```
                        ; ****************************************************************************
                        ; IBMBIO7.S (PCDOS 7.1 IBMBIO.COM) - RETRO DOS 5.0 by ERDOGAN TAN - 12/09/2023
                        ; ----------------------------------------------------------------------------
                        ; Last Update: 18/03/2025 - Retro DOS v5.0 (Modified PCDOS 7.1)
                        ; ----------------------------------------------------------------------------
                        ; Beginning: 26/12/2018 (Retro DOS 4.0), 01/10/2022 (Retro DOS 4.2)
                        ; ----------------------------------------------------------------------------
                        ; Assembler: NASM version 2.15
                        ; ----------------------------------------------------------------------------
                        ;      ((nasm ibmbio7.s -l ibmbio7.txt -o IBMBIO.COM -Z error.txt))
                        ;                                  -o IBMBIO7.BIN
                        ; ****************************************************************************

                        ; 12/09/2023 - Retro DOS v5.0 Kernel -dosbios- ('ibmbio7.s')
                        ; Modified from 'iosys6.s' (11/09/2023, Retro DOS v4.2 Kernel's IO.SYS) file
                        ; as below:
                        ;
                        ;    1) Retro DOS v4.2 IO.SYS is based on disassembled source code
                        ;   of MSDOS 6.21 IO.SYS, derived using MSDOS 6.0 source code.
                        ;
                        ;    2) Labels, names, comments, explanations and structure definitions
                        ;   about procedures and code details are almost entirely taken from
                        ;   the original MSDOS 6.0 source code, except for the details that
                        ;   Erdogan Tan personally experienced. Some of them are incompatible
                        ;   with PCDOS 7.1 code. But they have not been deleted to preserve
                        ;   the originality of the descriptions.)
                        ;
                        ;    3)'ibmbio7.s' contains the BIOSLOADER (MSLOADER) section located in
                        ;   the 1st 4 sectors of the IBMBIO.COM file on disk. This is a method
                        ;   from older DOS versions (3 sectors for MSDOS 6.22).
                        ;   The MSDOS/PCDOS boot sector code only reads these MSLOADER/BIOSLOADER
                        ;   sectors and transfers control to the MSLOADER/BIOSLOADER code.
                        ;   BUT!!! The Retro DOS v3 (& v5) boot sector code loads the entire
                        ;   MSDOS.SYS/PCDOS.SYS -combined- kernel file into memory at once.
                        ;   So, hence the Retro DOS boot sector code, 'retrodos5.s' file
                        ;       contains slightly different IO.SYS/IBMBIO.COM INITialization code
                        ;   than the original PCDOS/MSDOS. It does not include
                        ;   the MSLOADER/BIOSLOADER section. The 'retrodos5.s' and 'ibmbio7.s'
                        ;   files are almost identical except their INIT codes.)
                        ;
                        ; ('iosys6.s' has been converted to 'ibmbio7.s' and 'retrodos42.s' has been
                        ; converted to 'retrodos5.s'. 'ibmbio7.s' is IBMBIO.COM source code file
                        ; while 'retrodos5.s' is source code of Retro DOS v5 kernel file 'PCDOS.SYS'.
                        ; 'retrodos5.s' includes 'ibmdos7.bin' or IBMDOS.COM as binary file.)

                        ; ----------------------------------------------------------------------------

                        ; 09/12/2022 - Multisection binary file format (BIOSDATA & BIOSCODE sections)
                        ; 01/10/2022 - Erdogan Tan (Istanbul)

                        ;Note: This code is a part of Retro DOS 4.0 kernel source code
                        ;     (as included binary, 'IOSYS5.BIN')
                        ;     Equivalent of MSDOS 5.0 IO.SYS, BIOSCODE and BIOSDATA and SYSINIT
                        ;                                             (except MSLOAD code)

                        ;---- Retro DOS v2 (v3) boot sector loads RETRODOS.SYS (MSDOS.SYS)
                        ;     at 1000h:0000h and loader (initialization) part of RETRODOS kernel
                        ;     moves IO.SYS (DOSBIOSCODE & DOSBIOSDATA, 'IOSYS5.BIN') to 70h:0000h.
                        ;     Then SYSINIT code to the next segment (46Dh for original MSDOS 5.0)..
                        ;     SYSINIT code relocates itself and DOSBIOSCODE and MSDOS.SYS
                        ;     (MSDOS5.BIN) according to request/setting in 'config.sys' file.

                        ; ----------------------------------------------------------------------------

                        ; ----------------------------------------------------------------------------
                        ;
                        ; +-------------------------------------------------------------------------+
                        ; |   This file    has been generated by The Interactive Disassembler (IDA)    |
                        ; |        Copyright       (c) 2013 Hex-Rays, <support@hex-rays.com>         |
                        ; |                 Licensed to: Freeware version                           |
                        ; +-------------------------------------------------------------------------+
                        ;
                        ; ----------------------------------------------------------------------------

                        ;          .386
                        ;          .model flat

                        ; ============================================================================

                        ; 12/09/2023 - Erdogan Tan - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM

                        ;----------------------------------------------------------------------------

                        ; [[ Most of comments here are from the original MSDOS 6.0 source code ]]

                        ;----------------------------------------------------------------------------
                        ; Start of (PCDOS 7.1) IBMBIO.COM
                        ;----------------------------------------------------------------------------

                                    [ORG 0]               ; segment 0x0070h

                        ;============================================================================
                        ; IBMBIO.COM (IO.SYS) LOADER SECTION
                        ;============================================================================
                        ; 12/09/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
                        ; 09/12/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                        ;non-contiguous io.sys loader (msload) ((MSDOS 6.0 source: MSLOAD.ASM, 1991))

                        section .MSLOAD ; vstart=0 ;; .BIOSLOAD

                        ; ============================================================================

                        ; 09/12/2022
                        ; Comments are from MSDOS 6.0 MSLOAD.ASM (1991) & HEX-RAYS IDA disasm output

                        ; ============================================================================
                        ;   NOTE: The boot loader should be verifying that the first
                        ;     block of io.sys is, in fact, at cluster 2. This would be saving
                        ;     a whole lot of time during system debugging.
                        ;
                        ;============================================================================
                        ;
                        ;     for dos 4.00, msload program has been changed to allow:
                        ;     1. 32 bit calculation,
                        ;     2. reading a fat sector when needed, instead of reading the whole
                        ;        fat sectors at once. this will make the boot time faster,
                        ;        and eliminate the memory size limitation problem,
                        ;     3. solving the limitation of the file size (29 kb) of io.sys0,
                        ;     4. adding the boot error message. show the same boot error message
                        ;        and do the same behavior when the read operation of io.sys
                        ;        fails as the msboot program, since msload program is the
                        ;        extention of msboot program.
                        ;
                        ; ============================================================================
```

```
                                ;--------------------------------------------------------------------------
                                ;
                                ; M056 : Added RPL support, so that RPL's fake INT 13 code can be safe from
                                ;          SYSINIT & transient portion of COMMAND.COM
                                ;
                                ;--------------------------------------------------------------------------

                                           [ORG 0]              ; segment 0x0070h

                                START$:
00000000 EB45                              jmp    short SaveInputValues ; 13/09/2023
00000002 90                                nop    ; 13/09/2023
                                %if 0
                                ; 20/12/2022
                                ; 09/12/2022
                                ; --------------------------------------------------------------------------
                                SysVersion: dw 5                ; expected_version
                                ;MyStacks:  db 256 dup(0)       ; local stack
                                ; 22/12/2022
                                ; 20/12/2022
                                ;MyStacks:  dw 102 dup(0)       ; local stack
                                NumHeads:   dw 0                ; ...
                                ClusterSize:       db 2 dup(0)          ; ...
                                StartSecL:  dw 0                ; ...
                                StartSecH:  dw 0                ; ...
                                TempH:             dw 0                 ; for 32 bit calculation
                                TempCluster:       db 2 dup(0)          ; temporary place for cluster number
                                LastFatSector:     db 2 dup(0FFh)       ; fat sec # start from 1st FAT entry
                                SectorCount:       dw 0                 ; ...
                                SecPerFat:  dw 0                ; ...
                                HiddenSectorsL:    dw 0                 ; ...
                                HiddenSectorsH:    dw 0                 ; ...
                                BytesPerSec:       dw 0                 ; ...
                                ReservSectors:     db 2 dup(0)          ; ...
                                CurrentCluster:    db 2 dup(0)          ; ...
                                NextBioLocation: db 2 dup(0)            ; ...
                                FirstSectorL:      dw 0                 ; ...
                                FirstSectorH:      dw 0                 ; ...
                                TotalSectorsL:     dw 0                 ; max. number of sectors
                                TotalSectorsH:     dw 0                 ; ...
                                SecPerTrack:       db 2 dup(0)          ; ...
                                BootDrive:  db 0                ; ...
                                Fatsize:    db 0                ; ...
                                MediaByte:  db 0                ; ...
                                EndOfFile:  db 0                ; ...
                                OrgDasdPtr: db 4 dup(0)         ; ...
                                FatSegment: db 2 dup(0)         ; ...
                                SecPerCluster:     db 0                 ; ...
                                ; --------------------------------------------------------------------------
                                %endif

                                ; 13/09/2023 (Retro DOS v5)
                                ; 24/12/2022 (Retro DOS v4)
                                ; 23/12/2022
                                ; 20/12/2022
                                ; 09/12/2022
                                ; --------------------------------------------------------------------------
                                ;SysVersion:       dw 5                 ; expected_version
00000003 07                     SysVersionMajor: db 7     ; Retro DOS v5.0 (IBM PCDOS 7.1)
00000004 0A                     SysVersionMinor: db 10
00000005 0000                   ClusterSize:       dw 0
00000007 0000                   StartSecL:  dw 0
00000009 0000                   StartSecH:  dw 0
0000000B 0000                   TempH:             dw 0                 ; for 32 bit calculation
                                ;TempCluster:      dw 0
0000000D FFFF                   LastFatSectorL:    dw 0FFFFh            ; fat sec # start from 1st FAT entry
0000000F FFFF                   LastFatSectorH:    dw 0FFFFh            ; fat sec # start from 1st FAT entry
00000011 0000                   SectorCount:       dw 0
                                CurrentCluster:
                                           ; 06/10/2023
                                CurrentClusterL:
00000013 0000                              dw 0
                                CurrentClusterH:
00000015 0000                              dw 0    ; 13/09/2023 - HW of FAT32 cluster number
                                           ; 27/12/2023
                                FirstCluster:      ; 06/10/2023
00000017 0000                   FirstClusterL:     dw 0
00000019 0000                   FirstClusterH:     dw 0
                                           ;;
0000001B 0000                   BytesPerSec:       dw 0
0000001D 0000                   SecPerCluster:     dw 0
                                           ; 13/09/2023
0000001F 0000                   ReservSectors:     dw 0
00000021 0000                   NumFats:    dw 0
00000023 0000                   RootEntCnt: dw 0
00000025 0000                   SecPerTrack:       dw 0
00000027 0000                   NumHeads:   dw 0
00000029 0000                   HiddenSectorsL:    dw 0
0000002B 0000                   HiddenSectorsH:    dw 0
0000002D 0000                   TotalSectorsL:     dw 0                 ; max. number of sectors
0000002F 0000                   TotalSectorsH:     dw 0
00000031 0000                   FATSectorsL:       dw 0
00000033 0000                   FATSectorsH:       dw 0
00000035 0000                   RootClusterL:      dw 0
00000037 0000                   RootClusterH:      dw 0
                                           ;;
00000039 0000                   FirstSectorL:      dw 0
0000003B 0000                   FirstSectorH:      dw 0
0000003D 00                     BootDrive:  db 0
0000003E 00                     FatType:    db 0
0000003F 00                     MediaByte:  db 0
00000040 00                     EndOfFile:  db 0
00000041 00000000               OrgDasdPtr: dd 0
                                           ; 06/10/2023
                                ;FatStartSecL:     dw 0
                                ;FatStartSecH:     dw 0
00000045 0000                   FatSegment: dw 0
                                           ; 05/10/2023
                                ;NextBioLocation:
                                    ; 05/10/2023 (bp register will be used instead of [NextBioLocation])
                                           ;dw 0

                                           ; 13/09/2023

                                ; SaveInputValues
                                ; --------------------------------------------------------------------------
                                ; INPUT:     none
                                ;
                                ;   dl = int 13 drive number we booted from
                                ;   ch = media byte
                                ;   bx = first data sector (low) on disk (0-based)
                                ;   ds:si = original rom bios diskette parameter table.
                                ;
```

```
                        ; if an extended boot record, then ax will be the first data sector
                        ; high word. save ax and set FirstSectorH according to ax if it is an
                        ; extended boot record.
                        ;
                        ;   ax = first data sector (high) on disk ;
                        ; OUTPUT:
                        ;
                        ;   bx = first data sector on disk
                        ;
                        ;   MediaByte = input ch
                        ;   BootDrive = input dl
                        ;   FirstSectorL = input bx
                        ;   FirstSectorH = input ax, if an extended boot record.;j.k.
                        ;   TotalSectorsL = maximum sector number in this media ;j.k.
                        ;   TotalSectorsH = high word of the above
                        ;   HiddenSectorsL = hidden secotrs
                        ;   HiddenSectorsH
                        ;   ReservSectors = reserved sectors
                        ;   SecPerTrack = sectors/track
                        ;   NumHeads = heads/cylinder
                        ;
                        ;   ds = 0
                        ;   AX,DX,SI destroyed
                        ;
                        ; calls:     none
                        ; ---------------------------------------------------------------------------
                        ;FUNCTION:
                        ; save input information and bpb informations from the boot record.
                        ; ---------------------------------------------------------------------------

                        Sec9 equ 522h
                        ; 20/12/2022
                        DskAddr     equ 1Eh*4 ; 78h
                        ; 22/12/2022
                        ;StackPtr equ MyStacks+(NumHeads-MyStacks)

                        ; ---------------------------------------------------------------------------

                                                ; 13/09/2023
                                                ; (registers from PCDOS 7.1 boot sector)
                                                ; ss = 0
                                                ; sp = 7BE4h
                                                ;       [0:7BE4h] = ss:bx = 0:78h (1Eh vector)
                                                ;       [0:7BE8h] = ds:si = DSK_PARMS (INT 1Eh) table address
                                                ; bp = 7BECh
                                                ; ds = 0
                                                ; ax:bx = absolute disk address for cluster 2 (data start)
                                                ;       = dword/far ptr [0:7BFCh]
                                                ; es = ax
                                                ; dl = [BootDrv] = [7C40h] ; !FAT32 BPB!
                                                ; ch = [MediaByte] = [7C15h]
                                                ; ds:si = rom bios disk(ette) params table address (INT 1Eh)
                                                ;       = [0:7BE8h] = 0:7BECh
                                                ;       (ds:si is also in stack, at [0:7BE8h])
                                                ; 0:500h = root dir buffer (1st sector of the root dir)
                                                ; [0:7Eh] = disk(ette) params table address = 0:7BECh
                                                ;          (head settle time = 15ms)

                                        SaveInputValues:
                                                ; 13/09/2023 (Retro DOS v5 MSLOADER/BIOSLOADER)
 00000047 8CDF                                  mov     di, ds ; DSK_PARMS (INT 1Eh) table segment

                                                ; 24/12/2022 (Retro DOS v4 MSLOADER)
 00000049 0E                                    push    cs
 0000004A 1F                                    pop     ds
                                                ;mov    [cs:FirstSectorL], bx ; first data sector (low word)
                                                ;mov    [cs:MediaByte], ch
                                                ;mov    [cs:BootDrive], dl
                                                ; 13/09/2023
 0000004B 891E[3900]                            mov     [FirstSectorL], bx
 0000004F A3[3B00]                              mov     [FirstSectorH], ax
 00000052 891E[0700]                            mov     [StartSecL], bx ; **!**
 00000056 A3[0900]                              mov     [StartSecH], ax ; **!!**
 00000059 882E[3F00]                            mov     [MediaByte], ch
 0000005D 8816[3D00]                            mov     [BootDrive], dl

                                                ; 13/09/2023
                                                ; (PCDOS 7.1 MSLOAD:0058h)
                                                ;pop    si
                                                ;pop    ds
                                                ;       ; from BS code..
                                                ;       ; ss:sp = 0:7BE4h, bp = 7BECh
                                                ;       ; Clear stack and load disk parameters table in ds:si
                                                ;       ;
                                                ;       ; pop.. Original INT 1Eh vector address
                                                ;pop    si
                                                ;pop    ds
                                                ;       ; pop.. Original INT 1Eh disk table address

                                                ; 13/09/2023
                                                ; Note: DS:SI -from BS- points to DSK_PARMS (INT 1Eh) tbl addr
                                                ; (no need to pop/take address from stack)

                                                ;mov    sp, bp ; sp = 7BECh

                                                ; sp = 7BE4h

                                                ; 13/09/2023
                                                ; Save original (ROMBIOS) DSK_PARMS table address
 00000061 8936[4100]                            mov     [OrgDasdPtr], si   ; DSK_PARMS (INT 1Eh) tbl offset
 00000065 893E[4300]                            mov     [OrgDasdPtr+2], di ; DSK_PARMS (INT 1Eh) tbl segment

 00000069 31C9                                  xor     cx, cx          ; segment 0 (obviously)
 0000006B 8ED9                                  mov     ds, cx          ; ZERO
                                                ; 13/09/2023
 0000006D 8EC1                                  mov     es, cx
 0000006F 57                                    push    di
 00000070 BF2205                                mov     di, Sec9
 00000073 893E7800                              mov     [DskAddr], di  ; mov [78h], di ; 522h
 00000077 890E7A00                              mov     [DskAddr+2], cx ; mov [7Ah], cx ; 0
 0000007B 1F                                    pop     ds
 0000007C B10E                                  mov     cl, 14          ; (11+3 bytes for IBM rombios)
 0000007E FC                                    cld
 0000007F F3A4                                  rep movsb               ; copy table
                                                ; 20/12/2022
 00000081 8ED9                                  mov     ds, cx ; 0
                                                ; 23/12/2022
                                                ; es = 0
                                                ; ds = 0
                                                ; ss = 0

                                                ; 13/09/2023
                                                ;mov    cx, [051Ah]    ; LW of IBMBIO.COM (IO.SYS) first cluster
                                                ;mov    [cs:CurrentCluster], cx
```

```
                                ;mov    cx, [0514h]    ; HW of IBMBIO.COM (IO.SYS) first cluster
                                ;mov    [cs:CurrentCluster+2], cx
                        ; 24/12/2022
                        %if 0
                                mov     cx, [7C0Bh]    ; BootSector.ext_boot_bpb.BPB_bytespersector
                                mov     [cs:BytesPerSec], cx
                                mov     cl, [7C0Dh]    ; BootSector.ext_boot_bpb.BPB_sectorspercluster
                                mov     [cs:SecPerCluster], cl
                                mov     cx, [7C18h]    ; BootSector.ext_boot_bpb.BPB_sectorspertrack
                                mov     [cs:SecPerTrack], cx
                                mov     cx, [7C1Ah]    ; BootSector.ext_boot_bpb.BPB_heads
                                mov     [cs:NumHeads], cx
                                ;mov    cx, [7C16h]    ; BootSector.ext_boot_bpb.BPB_sectorsperfat
                                ;mov    [cs:SecPerFat], cx
                                ; 13/09/2023
                                mov     dx, [7C16h]    ; BootSector.ext_boot_bpb.BPB_sectorsperfat
                                ;mov    [cs:FATSectorsL], dx
                                mov     bl, [7C26h]    ; BS_BootSig ; (FAT12 and FAT16)
                                or      dx, dx ; **
                                jnz     short not_fat32
                                mov     bl, [7C42h]    ; BS_BootSig ; (FAT32)
                        not_fat32:
                                mov     cl, [7C10h]    ; BPB_NumFATs
                                mov     [cs:NumFats], cl
                                mov     cx, [7C11h]    ; BPB_RootEntCnt
                                mov     [cs:RootEntCnt], cx
                                ;
                                mov     cx, [7C0Eh]    ; BootSector.ext_boot_bpb.BPB_reservedsectors
                                mov     [cs:ReservSectors], cx
                                mov     cx, [7C1Ch]    ; BootSector.ext_boot_bpb.BPB_hiddensectors
                                mov     [cs:HiddenSectorsL], cx
                                mov     cx, [7C13h]    ; BootSector.ext_boot_bpb.BPB_totalsectors
                                mov     [cs:TotalSectorsL], cx

                                ; First of all, check if it the boot record is an extended one.
                                ; This is just a safe guard in case some user just "copy" the
                                ; 4.00 iosys.com to a media with a conventional boot record.
                                ; 22/12/2022
                                ;cmp    byte [7C26h], 29h ; ext_boot_signature
                                ; 13/09/2023
                                cmp     bl, 29h
                                jne     short Relocate ; old boot sector
                                                       ; no need to copy high words
                                mov     [cs:FirstSectorH], ax ; Start sector # of data, high word
                                mov     ax, [7C1Eh]    ; BPB_HiddSec+2
                                mov     [cs:HiddenSectorsH], ax
                                ; 10/12/2022
                                or      cx, cx
                                ;cmp    cx, 0          ; cx set already before (=totalsectors)
                                ; 22/12/2022
                                ;jnz    short Relocate
                                ; 13/09/2023
                                jnz     short not_big
                                mov     ax, [7C20h]    ; BootSector.ext_boot_bpb.BPB_bigtotalsectors
                                mov     [cs:TotalSectorsL], ax
                                mov     ax, [7C22h]    ; BootSector.ext_boot_bpb.BPB_bigtotalsectors+2
                                mov     [cs:TotalSectorsH], ax
                                ; 13/09/2023
                        not_big:
                                ;cmp    word [cs:FATsectorsL], 0
                                and     dx, dx ; **
                                jnz     short Relocate ; FAT12 or FAT16 fs

                                mov      cx, [7C24h]   ; BPB_FATSz32 ; FAT32 fs
                                mov      [cs:FATSectorsL], cx
                                mov      cx, [7C26h]   ; BPB_FATSz32+2
                                mov      [cs:FATSectorsH], cx
                                mov      cx, [7C2Ch]   ; BPB_RootClus
                                mov      [cs:RootClusterL], cx
                                mov      cx, [7C2Eh]    ; BPB_RootClus+2
                                mov      [cs:RootClusterH], cx
                        %endif
                                ; 13/09/2023 - Erdogan Tan - Istanbul
                                ;
                                ; Note: Boot signature check has been removed because
                                ;       it is not possible to start/run IBMBIO.COM
                                ;       if it would not be a valid FAT32 (or compatible) boot sector
                                ;       (input parameters and register contents would be wrong)

00000083 0E                     push    cs
00000084 07                     pop     es

                                ; 13/09/2023
00000085 BF[1700]               mov     di, FirstCluster
00000088 A11A05                 mov     ax, [051Ah]    ; LW of IBMBIO.COM (IO.SYS) first cluster
0000008B AB                     stosw                  ;  Initialize to this cluster
0000008C A11405                 mov     ax, [0514h]    ; HW of IBMBIO.COM (IO.SYS) first cluster
0000008F AB                     stosw

00000090 BE0B7C                 mov     si, 7C0Bh      ; boot sector's bpb, BytesPerSector
                                ;mov    di, BytesPerSec
00000093 A5                     movsw   ; BytesPerSec
00000094 A4                     movsb   ; SecPerCluster
00000095 47                     inc     di ; skip high byte of SecPerCluster word (it is 0)
00000096 A5                     movsw   ; ReservSectors
                                ; 13/09/2023
00000097 A4                     movsb   ; NumFats
00000098 47                     inc     di ; skip high byte of NumFats word (it is 0)
00000099 A5                     movsw   ; RootEntCnt
0000009A AD                     lodsw   ; TotalSectorsL
0000009B 50                     push    ax      ; save TotalSectorsL
0000009C AC                     lodsb   ; skip MediaByte
                                ; 13/09/2023
0000009D AD                     lodsw   ; FATSectorsL (Retro DOS 5) - SecPerFat (Retro DOS 4)
0000009E 89C2                   mov     dx, ax ; save BPB_FATSz16 into dx (it is 0 for FAT32 fs)
000000A0 A5                     movsw   ; SecPerTrack
000000A1 A5                     movsw   ; NumHeads
000000A2 A5                     movsw   ; HiddenSectorsL
000000A3 A5                     movsw   ; HiddenSectorsH
000000A4 58                     pop     ax      ; restore TotalSectorsL
                                ; si = 7C20h
                                ; di = offset TotalSectorsL
000000A5 09C0                   or      ax, ax ; 16 bit total sectors value
000000A7 7403                   jz      short big_total_sectors
000000A9 AB                     stosw   ; TotalSectorsL
                                        ; TotalsectorsH = 0
000000AA EB02                   jmp     short chk_fatsz_16

                        big_total_sectors:
                                ; BigTotalSecs - 32 bit total sectors value
000000AC A5                     movsw   ; BPB_TotSec32 (lw) -> TotalSectorsL
000000AD A5                     movsw   ; BPB_TotSec32 (hw) -> TotalSectorsH
                        chk_fatsz_16:
```

```
                                        ; 13/09/2023
                                        ; si = 7C24h
000000AE 09D2                           or      dx, dx ; **
000000B0 7405                           jz      short fat32_bs ; FAT32 boot sector
                                        ; FAT (FAT12 or FAT16) boot sector
000000B2 47                             inc     di  ; skip TotalsectorsH
000000B3 47                             inc     di
000000B4 92                             xchg    ax, dx  ; mov ax, dx
                                        ;stosw
000000B5 EB06                           jmp     short fat_bs

                                fat32_bs:
                                        ; FAT32 boot sector
000000B7 A5                             movsw   ; BPB_FATSz32 (lw) -> FATSectorsL
000000B8 A5                             movsw   ; BPB_FATSz32 (hw) -> FATSectorsH
000000B9 AD                             lodsw   ; skip BPB_ExtFlags
000000BA AD                             lodsw   ; skip BPB_FSVer
000000BB A5                             movsw   ; RootClusterL
000000BC AD                             lodsw   ; RootClusterH
                                fat_bs:
000000BD AB                             stosw   ; 13/09/2023

                                        ; 13/09/2023
000000BE 0E                             push    cs
000000BF 1F                             pop     ds

                                        ; 13/09/2023
                                        ; (PCDOS 7.1 - IBMBIO.COM - MSLOAD:0151h)

                                ; Relocate
                                ; ----------------------------------------------------------------------------
                                ;
                                ; NOTES:
                                ;
                                ;   Relocates the loader code to top-of-memory.
                                ;
                                ; INPUT:     none
                                ;
                                ; OUTPUT:    code and data relocated.
                                ;        AX,CX,SI,DI destroyed
                                ;
                                ; calls:     none
                                ; ----------------------------------------------------------------------------
                                ;
                                ; Determine the number of paragraphs (16 byte blocks) of memory.
                                ; this involves invoking the memory size determination interrupt,
                                ; which returns the number of 1k blocks of memory, and then
                                ; converting this to the number of paragraphs.
                                ; Find out whether RPL code is present at top of memory and modify the
                                ; available amount of memory in AX
                                ; leave the number of paragraphs of memory in ax.
                                ;
                                ; ----------------------------------------------------------------------------
                                ; copy code from start to top of memory.
                                ;
                                ; the length to copy is EndOfLoader
                                ;
                                ; jump to relocated code
                                ; ----------------------------------------------------------------------------

                                        ; 14/09/2023 - Retro DOS v5.0 BIOSLOADER/MSLOADER
                                        ;                              PCDOS 7.1
                                Relocate:
                                        ; 24/12/2022 - Retro DOS v4 (4.0 & 4.1 & 4.2) MSLOADER
                                        ;                              MSDOS 5.0 & 5.0+ & 6.22 (6.21)

                                        ;cld

000000C0 31F6                           xor     si, si
000000C2 89F7                           mov     di, si
000000C4 CD12                           int     12h             ; MEMORY SIZE -
                                                                ; Return: AX = number of contiguous 1K blocks of memory
000000C6 B106                           mov     cl, 6
000000C8 D3E0                           shl     ax, cl          ; Memory size in paragraphs

                                ;------ Check if an RPL program is present at TOM and do not tromp over it

                                        ; 10/12/2022
                                        ; ds = 0
                                        ; 24/12/2022
                                        ; ds = cs
                                        ;xor    bx, bx
                                        ;mov    ds, bx          ; ZERO
                                        ; 14/09/2023
000000CA 8EDE                           mov     ds, si ; 0

                                        ; 10/12/2022
000000CC 8B1EBC00                       mov     bx, [2Fh*4]     ; (Int 2Fh)
000000D0 8E1EBE00                       mov     ds, [2Fh*4+2]

                                        ;cmp    word ptr [bx+3], 'PR'
                                        ; 09/12/2022
000000D4 817F035250                     cmp     word [bx+3], 'RP' ; 'RPL'
000000D9 750F                           jnz     short Skip_RPL
000000DB 807F054C                       cmp     byte [bx+5], 'L'
000000DF 7509                           jnz     short Skip_RPL
000000E1 89C2                           mov     dx, ax          ; get TOM into DX
000000E3 B8064A                         mov     ax, 4A06h       ; (multMULT shl       8) + multMULTRPLTOM
000000E6 CD2F                           int     2Fh             ; Get new TOM from any RPL
000000E8 89D0                           mov     ax, dx
                                Skip_RPL:
                                        ; 24/12/2022
000000EA 0E                             push    cs
000000EB 1F                             pop     ds      ; 25/12/2022

000000EC B104                           mov     cl, 4
000000EE 8B16[1B00]                     mov     dx, [BytesPerSec] ; 24/12/2022
                                        ;mov    dx, [cs:BytesPerSec]
000000F2 D3EA                           shr     dx, cl
000000F4 42                             inc     dx
000000F5 29D0                           sub     ax, dx
000000F7 A3[4500]                       mov     [FatSegment], ax ; 24/12/2022
                                        ;mov    [cs:FatSegment], ax ; This will be used for fat sector
                                        ; 14/09/2023
                                        ;mov    dx, 5F0h ; 1520 (for PCDOS 7.1 IBMBIO.COM)
000000FA BA[8004]                       mov     dx, EndOfLoader ; loader size = 1520
000000FD D3EA                           shr     dx, cl
000000FF 42                             inc     dx
00000100 29D0                           sub     ax, dx
00000102 8EC0                           mov     es, ax  ; ES:DI -> place be relocated.
                                        ; 14/09/2023
                                        ; 22/12/2022
                                        ;dec    dx
                                        ;shl    dx, cl  ; convert paragraphs to bytes (*)
```

```
                                                        ; (stack pointer will be set to this offset)
                                        ; 24/12/2022
                                        ;push   cs
                                        ;pop    ds              ; DS:SI -> source

                                        ; 14/09/2023
00000104 B9[8004]                       mov     cx, EndOfLoader ; 1520 (for PCDOS 7.1 IBMBIO.COM)
00000107 F3A4                           rep movsb

00000109 06                             push    es              ; Far jump to relocated MSLOAD code
                                                                ; (via retf, far return)
0000010A B8[0F01]                       mov     ax, SetupStack
0000010D 50                             push    ax              ; Massage stack for destin of CS:IP
0000010E CB                             retf

                                ; ----------------------------------------------------------------------------

                                ; Start of relocated code
                                ;----------------------------------------------------------------------------
                                ;
                                ; Move the stack to just under the boot record and relocation area (0:7c00h)
                                ;

                                SetupStack:
                                        ; 22/12/2022
                                        ;mov    ax, cs          ; Start of relocated code
                                        ;mov    ss, ax
                                        ;;mov   sp, NumHeads    ; StackPtr offset
                                        ; 20/12/2022
                                        ;mov    sp, StackPtr    ; StackPtr offset

                                ; 22/12/2022
                                ; (set a temporary stack just above the relocated loader code)
                                ; ((instead of using/reserving 256 bytes of stack space in 'IO.SYS' file))

                                        ; 22/12/2022
                                        ; cs = loader segment (relocated)
                                        ; dx = loader size + stack space (*) -paragraph aligned-

                                        ; 14/09/2023
                                        ;;cli
                                        ;mov    ax, cs
                                        ;mov    ds, ax  ; 24/12/2022
                                        ;cli
                                        ;mov    ss, ax
                                        ;;mov   sp, dx  ; (*)
                                        ;sti

                                        ; 14/09/2023
                                        ; (PCDOS 7.1 IBMBIO.COM - MSLOAD:01A5h)
0000010F 8CC8                           mov     ax, cs
                                        ; 27/12/2023
00000111 8ED8                           mov     ds, ax
                                        ;
00000113 83E840                         sub     ax, 40h ; move ss to 400h backward for stack space
                                                        ; then set sp to the end of this stack space
00000116 8ED0                           mov     ss, ax
00000118 BC0004                         mov     sp, 400h
                                        ;add    ax, 40h ; ax = cs
                                        ;mov    ds, ax

                                ; FindClusterSize
                                ; ----------------------------------------------------------------------------
                                ;
                                ; INPUT:     bpb information in loaded boot record at 0:7c00h
                                ;
                                ; OUTPUT:
                                ;
                                ;   ds = 0
                                ;   ax = bytes/cluster
                                ;   bx = sectors/cluster
                                ;   si destroyed
                                ; calls:     none
                                ;----------------------------------------------------------------------------
                                ;
                                ; get bytes/sector from bpb
                                ;
                                ; get sectors/cluster from bpb
                                ;
                                ; bytes/cluster = bytes/sector * sector/cluster
                                ; ----------------------------------------------------------------------------

                                FindClusterSize:

                                ;for the time being just ASSUME the boot record is valid and the bpb is there.

                                        ; 14/09/2023
                                        ; 24/12/2022
                                        ; ds = cs
0000011B A1[1B00]                       mov     ax, [BytesPerSec]
                                        ;xor    bx, bx
                                        ;mov    bl, [SecPerCluster]   ; get sectors/cluster
                                        ;mul    bx
0000011E F726[1D00]                     mul     word [SecPerCluster]
                                        ; 14/09/2023
00000122 09D2                           or      dx, dx
00000124 7403                           jz      short CalcFatSize
00000126 E98E01                         jmp     ErrorOut

                                ; CalcFatSize
                                ; ----------------------------------------------------------------------------
                                ;
                                ; NOTES:
                                ;
                                ;   Determine if fat is 12 or 16 bit fat. 12 bit fat if floppy, read mbr
                                ;   to find out what system id byte is.
                                ;
                                ; INPUT:
                                ;
                                ; OUTPUT:
                                ;
                                ;   CS:FatSize = FAT_12_BIT or FAT_16_BIT
                                ;   all other registers destroyed
                                ;
                                ; ----------------------------------------------------------------------------

                                CalcFatSize:
                                        ; 14/09/2023 (Retro DOS v5, PCDOS 7.1 IBMBIO.COM LOADER)
                                        ; 24/12/2022 (Retro DOS v4, MSDOS 5.0-6.22 IO.SYS LOADER)

00000129 A3[0500]                       mov     [ClusterSize], ax ; cluster size in bytes

                                        ; 24/12/2022
                                        ; ds = cs
```

```
                                        ;mov    byte [Fatsize], 1; FAT_12_BIT (assume)
                                        ; 14/09/2023
0000012C C606[3E00]01                   mov     byte [FatType],1 ; FAT12

                                        ;mov    dx, [TotalSectorsH]
                                        ;mov    ax, [TotalSectorsL] ; DX:AX = total disk sectors
                                        ; 14/09/2023
00000131 A1[2F00]                       mov     ax, [TotalSectorsH]
00000134 8B1E[2D00]                     mov     bx, [TotalSectorsL] ; AX:BX = total disk sectors
                                ; 14/09/2023
                                %if 0
                                        sub     ax, [ReservSectors]

                                        sbb     dx, 0           ; DX:AX= Total available sectors

                                        push    ax
                                        push    dx

                                        mov     bx, [FATSectorsL]

                                        ;mov    cx, [FATSectorsH]
                                        ;push   ax
                                        ;push   dx
                                        ;;mov   al, [NumFats]
                                        ;;xor   ah, ah
                                        ;mov    ax, [NumFats]
                                        ;xchg   ax, cx
                                        ;mul    cx

                                        mov     ax, [FATSectorsH]
                                        mov     cx, [NumFats]  ; calculate total FAT sectors
                                        mul     cx
                                        xchg    ax, cx
                                        mul     bx
                                        add     cx, dx
                                        mov     bx, ax

                                        pop     dx
                                        pop     ax

                                        sub     ax, bx
                                        sbb     dx, cx          ; DX:AX = Total sectors - FAT sectors

                                        mov     bx, [RootEntCnt] ; Root directory entry count
                                        mov     cl, 4
                                        shr     bx, cl          ; BX = Total directory sectors
                                        sub     ax, bx
                                        sbb     dx, 0           ; DX:AX= Sectors in data area
                                %endif
                                        ; 14/09/2023
00000138 8B16[3900]                     mov     dx, [FirstSectorL]
0000013C 8B0E[3B00]                     mov     cx, [FirstSectorH]
                                        ; 04/10/2023
00000140 8916[0700]                     mov     [StartSecL], dx
00000144 890E[0900]                     mov     [StartSecH], cx
                                        ;
                                        ; ! here, cx:dx includes hidden sectors (partition start address) !
00000148 2B16[2900]                     sub     dx, [HiddenSectorsL]
0000014C 1B0E[2B00]                     sbb     cx, [HiddenSectorsH] ; cx:dx = start of data from boot sector

                                        ; 14/09/2023
00000150 29D3                           sub     bx, dx  ; total secs - start of data
00000152 19C8                           sbb     ax, cx
                                                        ; AX:BX= Sectors in data area
                                        ; 24/12/2022
00000154 8B0E[1D00]                     mov     cx, [SecPerCluster] ; *#*
                                        ; 14/09/2023
                                        ; bx = lw of data sector count
                                        ; ax = hw of data sector count
00000158 31D2                           xor     dx, dx
0000015A F7F1                           div     cx  ; *#*
                                        ; 24/12/2022
                                        ;;mov   [cs:TempH], ax ; AX = Total number of clusters (hw)
                                        ;mov    [TempH], ax
                                        ; 14/09/2023
0000015C 93                             xchg    ax, bx  ; ax = lw of data sector count ; 06/10/2023
                                                        ; bx = hw of cluster count
0000015D F7F1                           div     cx  ; *#*
                                        ; 14/09/2023
                                        ;mov    dx, [FirstCluster+2]
0000015F C606[3E00]0B                   mov     byte [FatType], 0Bh ; set FAT type to FAT32 (CHS type disk R/W)
                                        ;cmp    word [TempH], 0
                                        ;jne    short ReadInFirstCluster
00000164 09DB                           or      bx, bx  ; is cluster count > 65535 ?
00000166 7518                           jnz     short ReadInFirstCluster ; yes, it is (it must be) FAT32 fs
                                        ; 06/10/2023
00000168 83F8F6                         cmp     ax, 0FFF6h      ; FAT16 limit (65536-10)
0000016B 7313                           jnb     short ReadInFirstCluster ; FAT32
                                        ;
0000016D 891E[1900]                     mov     [FirstCluster+2], bx ; 0 ; (clear HW of FirstCluster)
                                        ;xor    dx, dx
00000171 C606[3E00]01                   mov     byte [FatType], 1 ; set FAT type to FAT12
                                        ; 06/10/2023
00000176 3DF60F                         cmp     ax, 0FF6h
                                        ;cmp    ax, 4086        ; 4096-10
00000179 7205                           jb      short ReadInFirstCluster ; 12 bit FAT
0000017B C606[3E00]04                   mov     byte [FatType], 4 ; set FAT type to FAT16

                                ; ReadInFirstCluster
                                ; ---------------------------------------------------------------------------
                                ;
                                ; NOTES: read the start of the clusters that covers at least IbmLoadSize
                                ;    fully.  for example, if sector/cluster = 2, and IbmLoadSize=3
                                ;    then we are going to re-read the second cluster to fully cover
                                ;    msload program in the cluster boundary.
                                ;
                                ; INPUT:
                                ;   IbmLoadSize - make sure this value is the same as the one in
                                ;             msboot program when you build the new version!!!!!
                                ;
                                ;   SecPerCluster
                                ;   ClusterSize
                                ;   FirstSectorL
                                ;   FirstSectorH
                                ;
                                ; OUTPUT: msload program is fully covered in a cluster boundary.
                                ;     ax = # of clusters we read in so far.
                                ;
                                ; calls:     ReadSectors
                                ; logic:
                                ;   ax; dx = IbmLoadSize / # of sector in a cluster.
                                ;   if dx = 0 then ok. (msload is in a cluster boundary.)
                                ;      else           (has to read (ax+1)th cluster to cover msload)
                                ;   read (ax+1)th cluster into the address after the clusters we
```

```
                                ;   read in so far.
                                ; ----------------------------------------------------------------------------

                                ; 09/12/2022
                                ; BiosStart equ 51Ah ; AX = IO.SYS starting cluster
                                ; IbmLoadSize equ 3  ; AX = Number sectors in MSLOAD
                                ; BiosOffset equ 700h ; Address where loader was read in

                                ReadInFirstCluster:
                                        ; 14/09/2023
00000180 8B16[1900]                     mov     dx, [FirstCluster+2]
00000184 A1[1700]                       mov     ax, [FirstCluster]
                                                        ; IBMBIO.COM First Cluster
                                                        ; Root dir buffer at 500h (segment=0)
                                                        ; IBMBIO.COM first cluster ptr at 51Ah
                                                        ; high word of cluster is at 514h
                                        ; 14/10/2023 (!*!)
                                        ;; a cluster number start from 2
                                        ;; convert it to (correct) cluster index number
                                        ;sub    ax, 2
                                        ;sbb    dx, 0
                                        ;       ; DX:AX = zero based cluster number (cluster index)

00000187 8916[1500]                     mov     [CurrentClusterH], dx
0000018B A3[1300]                       mov     [CurrentClusterL], ax ; Initialize to this cluster

                                        ; 24/12/2022
                                        ; (PCDOS 7.1 IBMBIO.COM - MSLOAD:0255h) ; 04/10/2023 ('mov ax,3')
                                        ;;mov   ax, IbmLoadSize
                                        ;mov    ax, 3           ; Load the 3rd and other IO.SYS      sectors
                                        ; 04/10/2023  ; **
                                        ; (Windows ME IO.SYS - MSLOAD:01E4h)
0000018E B80400                         mov     ax, 4 ; **      ; Load the 4rd and other IO.SYS      sectors

                                        ; 14/09/2023
                                        ;div    byte [SecPerCluster]
00000191 F6F1                           div     cl ; *#*
                                                        ; AL = total cluster read in
                                                        ; AH = remaining sectors in last cluster

                                        ; (Note: PCDOS 7.1 bs loads 1st 4 sectors of IBMBIO.COM)
                                        ; If cluster size > 3, al = 0, ah <> 0
                                        ; If cluster size = 2, al = 1, ah = 1
                                        ; If cluster size = 1, al = 3, ah = 0
                                        ; If ah = 0, nothing remaining in last cluster

                                        ; 14/10/2023
00000193 BE7000                         mov     si, 70h ; ++*
00000196 8EC6                           mov     es, si  ; ++    ; ES = BIOSDATA (IO.SYS DATA) segment

                                        ; 14/09/2023
                                        ;cmp    ah, 0
                                        ; 10/12/2022
00000198 20E4                           and     ah, ah
                                        ;cmp    ah, 0
0000019A 742F                           jz      short SetNextClusterNum       ; next cluster
                                                ; 04/10/2023 ; **
                                                ; If AH=0
                                                ;    and if CL=1, AL=4
                                                ;    and if CL=2, AL=2 (?)
                                                ;    and if CL=4, AL=1
                                                ; If AH>0
                                                ;    AL=0 and AH=4

                                        ; 04/10/2023   ; **
                                        ; al = 0
                                        ;xor    ah, ah ; 0
                                        ;push   ax      ; (*)   ; AX = total clusters in the loader
                                                                ; already read in
                                ; 14/09/2023
                                ; 24/12/2022
                                %if 0
                                        mov     cx, [FirstSectorL]    ; Put starting sector of disk data
                                        mov     [StartSecL], cx       ; area in StartSecH:StartSecL
                                        mov     cx, [FirstSectorH]
                                        mov     [StartSecH], cx
                                        mul     byte [cs:SecPerCluster]
                                        add     [StartSecL], ax       ; Add number of sectors already loaded
                                        adc     word [StartSecH], 0   ; to start sector
                                        ;mov    dx, [FirstCluster+2]
                                        ;mov    ax, [FirstCluster]
                                        ;sub    ax, 2
                                        ;sbb    dx, 0
                                        mov     dx, [CurrentClusterH] ; IBMBIO.COM 1st cluster (index)
                                        mov     ax, [CurrentClusterL] ; (zero based cluster number)

                                        xor     bx, bx
                                        mov     bl, [SecPerCluster]
                                        mul     bx              ; DX:AX = logical start sector
                                        add     [StartSecL], ax
                                        adc     [StartSecH], dx
                                                                ; abs start sector for next read of
                                                                ; the rest of the last loader cluster
                                        pop     ax      ; (*) number of clusters already loaded
                                                        ; (0 or 1)
                                                        ; (Note: if al=0, the 1st 4 sectors of the 1st cluster
                                                        ; will be loaded again! -PCDOS 7.1-)
                                        push    ax
                                        mul     word [ClusterSize]
                                        ;mov    di, BiosOffset
                                        mov     di, 700h        ; IBMBIO.COM (IO.SYS) loading addr (segment = 0)
                                        add     di, ax
                                        xor     ax, ax
                                        mov     es, ax          ; ES = segment 0
                                        mov     al, [SecPerCluster]
                                                                ; Read in the entire last cluster
                                        mov     [SectorCount], ax
                                        call    ReadSectors
                                        pop     ax              ; AX = total clust read       by boot loader
                                        inc     ax              ; AX = total clust read       in now
                                SetNextClusterNum:              ; ...
                                        inc     ax              ; AX = total clusters read in based 2
                                        add     [CurrentClusterL], ax
                                        adc     [CurrentClusterH], 0
                                        dec     ax              ; CurrentCluster = Last       cluster read
                                                                ; AX = number of clusters loaded
                                %endif
                                        ; 04/10/2023 ; **
                                        ; ah=4 & al=0
0000019C 88C8                           mov     al, cl ; ** (SecPerCluster)
0000019E 28E0                           sub     al, ah ; ** (remain sectors to read in the cluster)
000001A0 A2[1100]                       mov     [SectorCount], al ; ** (spc-4)

                                        ; 24/12/2022
```

```
                                        ; ds = cs
                                        ;mov    cx, [FirstSectorL] ; Put starting sector of disk data
                                        ;mov    [StartSecL], cx    ; area in StartSecH:StartSecL
                                        ;mov    cx, [cs:FirstSectorH]
                                        ;mov    [StartSecH], cx
                                        ; [StartSecL] = [FirstSectorL] ; **!**
                                        ; [StartSecH] = [FirstSectorH] ; **!**

                                        ; 24/12/2022
                                        ; cx = [SecPerCluster]  ; *#*

                                        ; 04/10/2023
                                        ;; ax = 0 (cluster size > 3) or ax = 1 (cluster size = 2)
                                        ; al = 0 (cluster size > 4) and ah > 0  ; (as Win ME IO.SYS)
                                        ; cx = sectors per cluster (ch = 0)

                                        ; 04/10/2023 ; **
                                        ; cx = sectors per cluster (ch = 0)

                                ;       ;or     al, al ; *
                                ;       ;jz     short rfc_1 ; al = 0 ; *
                                        ; al = 1

                                        ;;mul   byte [SecPerCluster]
                                        ;mul    cl ; *#*
                                        ;add    [StartSecL], ax ; Add number of sectors already loaded
                                        ;adc    word [StartSecH], 0 ; to start sector
                                        ; 04/10/2023
                                        ;add    [StartSecL], cx ; * (AL=1, CL*AL=CL, CH=0)
                                        ;adc    word [StartSecH], 0 ; *

                                ;rfc_1: ; * ; 04/10/2023
                                        ;mov    ax, [51Ah] ; AX = [51Ah] = IO.SYS 1st clust
                                        ;dec    ax
                                        ;dec    ax
                                        ; 14/10/2023
                                        ;mov    ax, [CurrentClusterL] ; *##*
                                        ;; ax = word [51Ah] - 2
                                        ; 04/10/2023
                                        ;mov    dx, [CurrentClusterH]
000001A3 A1[1500]                       mov     ax, [CurrentClusterH] ; *###*

                                        ;;xor   bx, bx
                                        ;;mov   bl, [SecPerCluster]
                                        ;mov    bx, [SecPerCluster]
                                        ;mul    bx              ; DX:AX = logical start sector
                                        ; 04/10/2023
                                        ;mul    cx  ; [SecPerCluster]  ; *#*
                                        ; 32 bit multiplication (HHLL*SPC)
                                        ; 14/10/2023
                                        ;push   ax ; Current Cluster LW
                                        ;mov    ax, dx ; Current Cluster HW (HH) ; *###*

000001A6 F7E1                           mul     cx ; (HH*SPC) ; (result: dx is -must be- zero)
000001A8 91                             xchg    ax, cx
                                        ; 14/10/2023
                                        ;pop    dx ; Current Cluster LW       (LL)
                                        ;mul    dx  ; LL*SPC
000001A9 F726[1300]                     mul     word [CurrentClusterL] ; *##*
000001AD 01CA                           add     dx, cx ; (add lw of HH*SPC)

                                        ; 04/10/2023 ; **
000001AF 83C004                         add     ax, 4 ; ** ; IbmLoadSize (Win ME BS's IO.SYS read count)
000001B2 83D200                         adc     dx, 0 ; **

000001B5 0106[0700]                     add     [StartSecL], ax
000001B9 1116[0900]                     adc     [StartSecH], dx
                                                                ; abs start sector for next read of
                                                                ; the rest of the last loader cluster
                                        ; 04/10/2023 ; **
                                        ; (number of clusters already -complete- loaded = 0)
                                        ;pop    ax      ; (*) number of clusters already loaded
                                        ;push   ax

                                        ; 04/10/2023 ; **
                                        ;;mul   word [ClusterSize]
                                        ; 14/10/2023
                                        ;mov    ax, [BytesPerSec]
                                        ;shl    ax, 2 ; * 4 (4 sectors already loaded)

                                        ;;mov   di, BiosOffset
                                        ;mov    di, 700h        ; IO.SYS offset       (segment = 0)
                                        ;add    di, ax
                                        ; 14/10/2023
000001BD 8B3E[1B00]                     mov     di, [BytesPerSec]
000001C1 C1E702                         shl     di, 2 ; * 4 (4 sectors already loaded)
                                        ;add    di, 700h ; ++
                                        ; di = buffer offset

                                        ; 04/10/2023 ; **
                                        ;xor    ax, ax
                                        ;mov    es, ax          ; ES = segment 0
                                        ;; dx = 0 ; **
                                        ;mov    es, dx ; 0
                                        ; 14/10/2023
                                        ; cx = 0
                                        ;mov    es, cx ; 0 ; ++
                                        ; es = buffer segment = 0

                                        ; 14/10/2023
                                        ; es = si = 70h ; ++*
                                        ;mov    si, 70h ; ++
                                        ;mov    es, si  ; ++
                                        ; es:di = 70h:800h

                                        ; 24/12/2022
                                        ;mov    al, [SecPerCluster]
                                        ;                       ; Read in the entire last cluster
                                        ;mov    [SectorCount], ax
                                        ; 14/10/2023 ; **
                                        ;mov    [SectorCount], cx ; [SecPerCluster]  ; *#*

000001C4 E87F00                         call    ReadSectors

                                        ; 04/10/2023 ; *
                                        ;;pop   ax              ; AX = total clust read      by boot loader
                                        ;;inc   ax              ; AX = total clust read      in now
                                        ; 04/10/2023
                                        ;mov    ax, 1   ; 1 cluster loaded
                                ;SetNextClusterNum:
                                        ; 14/10/2023
000001C7 89F8                           mov     ax, di
                                        ; ax = loaded (IBMBIO.COM or IO.SYS) byte count
000001C9 EB0F                           jmp     short SaveLoadedBios2
```

```
                                SetNextClusterNum:
                                            ; 14/10/2023 (!*!)
                                            ;inc    ax
000001CB 48                                 dec     ax ; (4 clusters -> +3, 1 cluster -> +0)
000001CC 0106[1300]                         add     [CurrentClusterL], ax  ; ah = 0
000001D0 8316[1500]00                       adc     word [CurrentClusterH], 0
                                                            ; CurrentCluster = Last       cluster (loaded)
000001D5 40                                 inc     ax
                                            ;dec    ax
                                                            ; AX = number of clusters loaded

                                ; SaveLoadedBios
                                ; ---------------------------------------------------------------------------
                                ;
                                ; NOTES:
                                ;
                                ;   Determine how much of iosys was loaded in when the loader was loaded
                                ;   by the boot record (only the portion that is guaranteed to be contiguous)
                                ;
                                ; INPUT:
                                ;   AX:Total cluster already read in (loader & bios)
                                ;   CS:CurrentCluster = number of clusters used for loader+2
                                ;
                                ; OUTPUT:
                                ;   ES = 70h
                                ;   DI = next offset to load iosys code
                                ;   AX,BX,CX,DX,SI destroyed
                                ;
                                ;   CS:NextBioLocation = di on output
                                ;   CS:last_cluster = last cluster loaded
                                ;
                                ; calls:     none
                                ;
                                ; ---------------------------------------------------------------------------
                                ;
                                ; Multiply cluster * cluster size in bytes to get total loaded for msload
                                ;
                                ; Subtract total_loaded - (EndOfLoader) to get loaded io.sys in last cluster
                                ;
                                ; Relocate this piece of iosys down to 70:0
                                ;
                                ; ---------------------------------------------------------------------------

                                SaveLoadedBios:
                                            ; 14/10/2023
                                            ;push   ds

                                            ; 24/12/2022
                                            ; ds = cs
                                            ; ax = number of loaded clusters
000001D6 F726[0500]                         mul     word [ClusterSize]
                                            ;mul    word [cs:ClusterSize]
                                                                    ; Get total bytes loaded by
                                                                    ; this is always < 64k, so
                                                                    ; lower 16 bits ok
                                            ; 14/10/2023
                                            ; ax = [clusterSize] * (loaded cluster count)
                                SaveLoadedBios2:
                                            ; 14/10/2023
000001DA 1E                                 push    ds

                                            ; 14/10/2023
                                            ;sub    ax, EndOfLoader ; (OFFSET EndOfLoader)-(OFFSET Start)
000001DB BE[8004]                           mov     si, EndOfLoader
000001DE 29F0                               sub     ax, si
000001E0 89C1                               mov     cx, ax

                                            ; 14/10/2023
                                            ;mov    ax, 70h         ; Segment at 70h
                                            ;mov    ds, ax
                                            ;mov    es, ax
                                            ; es = 70h
000001E2 06                                 push    es
000001E3 1F                                 pop     ds

                                            ;mov    si, EndOfLoader
000001E4 31FF                               xor     di, di
000001E6 F3A4                               rep movsb               ; Relocate this code to 0070h:0000h

                                            ;mov    [NextBioLocation], di
                                            ;;mov   [cs:NextBioLocation], di
                                            ; 05/10/2023
                                            ;mov    [NextBioLocation], di
000001E8 89FD                               mov     bp, di

                                            ; es:di = (the next) buffer address for next read

000001EA 1F                                 pop     ds              ; Save where location for next read

                                ; GetContigClusters
                                ; ---------------------------------------------------------------------------
                                ;
                                ; NOTES: go find clusters as long as they are contiguous
                                ;
                                ;
                                ; INPUT:
                                ;   CS:NextBioLocation
                                ;   CS:
                                ;
                                ; OUTPUT:
                                ;
                                ; calls: GetNextFatEntry
                                ; ---------------------------------------------------------------------------
                                ;
                                ;Set CS:SectorCount to sectors per cluster
                                ;
                                ;Call GetNextFatEntry to get next cluster in file
                                ;
                                ;Call check_for_eof
                                ;
                                ;if (nc returned)
                                ;
                                ;   {call GetNextFatEntry
                                ;
                                ;    if (new cluster is contig to old cluster)
                                ;   {add sectors per cluster to CS:SectorCount
                                ;
                                ;    call check_for_eof
                                ;
                                ;    if (nc returned)
                                ;
                                ; ---------------------------------------------------------------------------

                                ; 09/12/2022
```

```
                        ; END_OF_FILE equ 0FFh
                        ; DosLoadSeg equ 70h

                        GetContigClusters:

                        ; 24/12/2022
                        %if 0
                                xor     ah, ah
                                mov     al, [cs:SecPerCluster]; Assume we will get one cluster
                                mov     [cs:SectorCount], ax  ; Sector count = sectors in 1 cluster
                                push    word [cs:SectorCount]
                                call    GetNextFatEntry               ; Returns next cluster to read in AX
                                pop     word [cs:SectorCount]
                                mov     word [cs:CurrentCluster], ax ; Update the last one found
                                cmp     byte [cs:EndOfFile], 0FFh ; END_OF_FILE
                                jz      short GoToBioInit
                                xor     dx, dx
                                ;sub    ax, 2                 ; Zero base the cluster
                                ; 10/12/2022
                                dec     ax
                                dec     ax
                                xor     ch, ch
                                mov     cl, [cs:SecPerCluster]
                                mul     cx                    ; How many sectors (before next cluster)
                                add     ax, [cs:FirstSectorL] ; See where the data sector starts
                                adc     dx, [cs:FirstSectorH]
                                mov     [cs:StartSecL], ax    ; Save it (used by ReadSectors)
                                mov     [cs:StartSecH], dx
                                mov     di, [cs:NextBioLocation] ; Get where to put code
                                push    word [cs:SectorCount] ; Save how many sectors
                                ;mov    ax, DosLoadSeg
                                mov     ax, 70h
                                mov     es, ax
                                call    ReadSectors
                                pop     ax                    ; Get back total sectors read in
                                mul     word [cs:BytesPerSec] ; Get number of bytes we loaded
                                add     [cs:NextBioLocation], ax ; Point to where to load next
                                jmp     short GetContigClusters

                        %endif
                                ; 24/12/2022
                                ; ds = cs
000001EB A1[1D00]               mov     ax, [SecPerCluster]   ; Assume we will get one cluster
000001EE A3[1100]               mov     [SectorCount], ax     ; Sector count = sectors in 1 cluster
                                ;push   word [SectorCount]
000001F1 50                     push    ax

000001F2 E85401                 call    GetNextFatEntry               ; Returns next cluster to read in AX

                                ;pop    word [SectorCount]
                                ; 05/10/2023
000001F5 59                     pop     cx ; sc = spc

000001F6 A3[1300]               mov     [CurrentClusterL], ax ; Update the last one found, lw
000001F9 893E[1500]             mov     [CurrentClusterH], di ; hw

000001FD 803E[4000]FF           cmp     byte [EndOfFile], 0FFh; END_OF_FILE
00000202 742A                   je      short GoToBioInit ; 23/12/2022

                                ; 22/12/2022
                                ;xor    dx, dx ; * (not required)
                                ; 10/12/2022
                                ;;sub   ax, 2                 ; Zero base the cluster
                                ;dec    ax
                                ;dec    ax
                                ; 14/10/2023
                                ;xor    dx, dx
00000204 83E802                 sub     ax, 2                 ; Zero base the cluster (32 bit as di:ax)
00000207 83DF00                 sbb     di, 0

                                ;; 24/12/2022
                                ;; ax = cluster index
                                ;;mov   cx, [SecPerCluster]
                                ;;mul   cx ; *                 ; How many sectors (before next cluster)
                                ; 04/10/2023
                                ;;mul   word [SecPerCluster]
                                ;mul    cx
                                ; 04/10/2023
0000020A 97                     xchg    di, ax                ; 32 bit multiplication
0000020B F7E1                   mul     cx                    ; (dx:ax)*cx
0000020D 97                     xchg    ax, di
0000020E F7E1                   mul     cx
00000210 01FA                   add     dx, di

00000212 0306[3900]             add     ax, [FirstSectorL]    ; See where the data sector starts
00000216 1316[3B00]             adc     dx, [FirstSectorH]

0000021A A3[0700]               mov     [StartSecL], ax              ; Save it (used by ReadSectors)
0000021D 8916[0900]             mov     [StartSecH], dx

                                ; 05/10/2023
                                ;mov    di, [NextBioLocation]
                                mov     di, bp
00000221 89EF                   mov     [SectorCount], cx
00000223 890E[1100]             ; es = 70h

                                ; es:di = (current) buffer address for (current) read

00000227 E81C00                 call    ReadSectors
                                                ; ES:DI = (the next) buffer address for next read
                                ; 05/10/2023
                                ;mov    [NextBioLocation], di
0000022A 89FD                   mov     bp, di

0000022C EBBD                   jmp     short GetContigClusters

                        ; ----------------------------------------------------------------------------

                        ; GoToBiosInit
                        ; ----------------------------------------------------------------------------
                        ;
                        ; NOTES:
                        ;
                        ;  Set up required registers for iosys, then jump to it (70:0)
                        ;
                        ; INPUT:     none
                        ;
                        ;   CS:MediaByte = media byte
                        ;   CS:BootDrive = int 13 drive number we booted from
                        ;   CS:FirstSectorL = first data sector on disk (low) (0-based)
                        ;   CS:FirstSectorH = first data sector on disk (high)
                        ;
                        ; OUTPUT:
                        ;
```

```
                                ;   required by msinit
                                ;   DL = int 13 drive number we booted from
                                ;   CH = media byte
                                ;   BX = first data sector on disk (0-based)
                                ;   AX = first data sector on disk (high)
                                ;   DI = sectors/fat for the boot media.
                                ;
                                ; calls:     none
                                ; -----------------------------------------------------------------------
                                ;
                                ; set up registers for msinit then do far jmp
                                ;
                                ; -----------------------------------------------------------------------

                                GoToBioInit:

                                ; 05/10/2023 - Retro DOS v5.0 IBMBIO.COM (PCDOS 7.1)
                                %if 0
                                        ; 24/12/2022
                                        ; ds = cs
                                        ;mov    ch, [cs:MediaByte]
                                        ;mov    dl, [cs:BootDrive]
                                        ;mov    bx, [cs:FirstSectorL]
                                        ;mov    ax, [cs:FirstSectorH]

                                        mov     ch, [MediaByte] ; Restore regs required for msint
                                        mov     dl, [BootDrive] ; Physical drv number we booted from.
                                        mov     bx, [FirstSectorL] ; AX:BX = first data sector of disk
                                        mov     ax, [FirstSectorH]
                                %endif

                                ; 05/10/2023 - Retro DOS v5.0 IBMBIO.COM (PCDOS 7.1)
                                %if 1
                                        ; 05/10/2023
                                        ; (PCDOS 7.1 IBMBIO.COM - MSLOAD:034Ah)
0000022E 8A2E[3F00]                     mov     ch, [MediaByte]         ; Set up required registers for iosys,
                                                                        ;  then jump to it (70:0)
                                                                        ;
                                                                        ; Restore regs required for msint
00000232 8A16[3D00]                     mov     dl, [BootDrive]         ; Physical drv number we booted from
00000236 8B1E[3900]                     mov     bx, [FirstSectorL]
0000023A A1[3B00]                       mov     ax, [FirstSectorH]
                                                                        ; bx:ax = first data sector of disk
0000023D C536[4100]                     lds     si, [OrgDasdPtr]
                                                                        ; Set ds:si to Original INT 1Eh disk(ette)
                                                                        ; table address and then push disk table
                                                                        ; address and INT 1Eh vector to stack
                                                                        ; (set stack content just as at the start
                                                                        ;  of MSLOAD)
                                        ; 05/10/2023
                                        ; following pushes are not necessary..
                                        ; PCDOS 7.1 BIOSDATA init procedure does not pop the pushed
                                        ; registers here and also it doesn't use the di value here
                                        ; (but ds:si is used)
                                        ;
                                        ;push   ds              ; INT 1Eh original table segment
                                        ;push   si              ; INT 1Eh original table offset
                                        ;xor    di, di          ; 0
                                        ;push   di              ; INT 1Eh vector segment
                                        ;mov    di, 78h         ; 1Eh*4 = 78h
                                        ;push   di              ; INT 1Eh vector offset
                                %endif
                                        ; 05/10/2023
                                        ;mov    bp, sp  ; not necessary
                                                        ; (BIOSDATA init doesn't use the bp value here)

00000241 EA00007000                     jmp     70h:0   ; Far jump to IoSysAddr      (DOSBIOS)

                                ; =============== S U B   R O U TI N E =======================================

                                ; 05/10/2023 - Retro DOS v5.0 IBMBIO.COM (Modified PCDOS 7.1)
                                ; (PCDOS 7.1 IBMBIO.COM - MSLOAD:036Bh)

                                %if 0
                                check_int13h_extensions:
                                        push    ax
                                        push    dx
                                        xor     ax, ax
                                        push    ax              ; zero (buffer offset 24)
                                                                ; (bytes per sector)
                                        mov     bx, sp
                                        sub     sp, 20
                                        push    ax              ; info flags
                                        mov     ax, 26          ; Result buffer size
                                        push    ax
                                        mov     si, sp
                                        mov     dl, [BootDrive]
                                        mov     ah, 48h
                                        push    ds
                                        push    ss
                                        pop     ds
                                        cmp     dl, 0
                                        jge     short not_hard_disk
                                        int     13h     ; DISK - IBM/MS Extension - GET DRIVE PARAMETERS
                                                        ; (DL - drive, DS:SI - buffer)
                                not_hard_disk:
                                        pop     ds
                                        mov     sp, bx
                                        pop     ax              ; bytes per sector, buffer offset 24
                                        jc      short int13h_ext_err
                                        cmp     ax, 512
                                        je      short int13_ext_ok
                                        stc
                                int13h_ext_err:
                                int13_ext_ok:
                                        pop     dx
                                        pop     ax
                                        retn
                                %endif

                                ; =============== S U B   R O U TI N E =======================================

                                ; ReadSectors
                                ; -----------------------------------------------------------------------
                                ; notES:
                                ;
                                ;  read in the CS:SectorCount number of sectors at ES:di
                                ;
                                ;
                                ; INPUT:
                                ;
                                ;   DI = OFFSET of start of read
                                ;   ES = segment of read
                                ;   CS:SectorCount = number of sectors to read
```

```
                        ;   CS:StartSecL = starting sector (low)
                        ;   CS:StartSecH = starting sector (high)
                        ;   following is bpb info that must be setup prior to call
                        ;   CS:NumHeads
                        ;   CS:number_of_sectors
                        ;   CS:BootDrive
                        ;   CS:SecPerTrack
                        ;
                        ; OUTPUT:
                        ;
                        ;   AX,BX,CX,DX,SI,DI destroyed
                        ; ---------------------------------------------------------------------------
                        ; divide start sector by sectors per track
                        ; the remainder is the actual sector number, 0 based
                        ;
                        ; increment actual sector number to get 1 based
                        ;
                        ; the quotient is the number of tracks - divide by heads to get the cyl
                        ;
                        ; the remainder is actual head, the quotient is cylinder
                        ;
                        ; figure the number of sectors in that track, set al to this
                        ;
                        ; do the read
                        ;
                        ; if error, do reset, then redo the int 13h
                        ;
                        ; if successful read, subtract # sectors read from SectorCount, add to logical
                        ; sector, add #sectors read * BytesPerSec to bx;
                        ;
                        ; if SectorCount <> 0 do next read
                        ; ---------------------------------------------------------------------------

                        ; 05/10/2023 - Retro DOS v5.0 IBMBIO.COM (Modified PCDOS 7.1)

                                    ; 24/12/2022
                                    ; 22/12/2022
                        ReadSectors:
00000246 B90500                     mov     cx, 5                   ; 5 retries

                                    ; Convert a logical sector into track/sector/head. AX has the
                                    ; logical sector number
                        TryRead:
                                    ; 24/12/2022
                                    ; ds = cs
00000249 51                         push    cx              ; (*)
                                    ;mov    ax, [cs:StartSecL]      ; Get starting sector
                                    ;mov    dx, [cs:StartSecH]
0000024A A1[0700]                   mov     ax, [StartSecL]                 ; Get starting sector
0000024D 8B16[0900]                 mov     dx, [StartSecH]
00000251 50                         push    ax              ; (**)
                                    ;;;

                                    ; 05/10/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                    ;call   check_int13h_extensions
                                    ;jc     short chs_read
                                    ; ----------------------
                                    ; 06/10/2023
                        check_int13h_extensions:
00000252 8A1E[3D00]                 mov     bl, [BootDrive]
00000256 08DB                       or      bl, bl
00000258 7953                       jns     short chs_read  ; not hard disk
                                    ; bl >= 80h
0000025A 52                         push    dx ; ***
                                    ;mov    dl, [BootDrive]
0000025B 88DA                       mov     dl, bl
0000025D 50                         push    ax ; ****
0000025E 31C0                       xor     ax, ax
00000260 50                         push    ax              ; zero (buffer offset 24)
                                                            ; (bytes per sector)
00000261 89E3                       mov     bx, sp
00000263 83EC14                     sub     sp, 20
00000266 50                         push    ax              ; info flags
00000267 B81A00                     mov     ax, 26          ; Result buffer size
0000026A 50                         push    ax
0000026B 89E6                       mov     si, sp
0000026D B448                       mov     ah, 48h
0000026F 1E                         push    ds
00000270 16                         push    ss
00000271 1F                         pop     ds
00000272 CD13                       int     13h     ; DISK - IBM/MS Extension - GET DRIVE PARAMETERS
                                                    ; (DL - drive, DS:SI - buffer)
00000274 1F                         pop     ds
00000275 89DC                       mov     sp, bx
00000277 58                         pop     ax              ; bytes per sector, buffer offset 24
00000278 7203                       jc      short int13h_ext_err
0000027A 3D0002                     cmp     ax, 512
                        int13h_ext_err:
0000027D 58                         pop     ax ; ****
0000027E 5A                         pop     dx ; ***
0000027F 752C                       jne     short chs_read
                                    ; ----------------------
                        lba_read:
                                    ;xor    si, si          ; LBA read
                                    ;push   si              ; 0
                                    ;push   si              ; 0
00000281 31DB                       xor     bx, bx          ; LBA read
00000283 53                         push    bx              ; 0
00000284 53                         push    bx              ; 0
00000285 52                         push    dx
00000286 50                         push    ax              ; 0:0:dx:ax = start sector (8 bytes)
00000287 06                         push    es
00000288 57                         push    di              ; memory buffer address (seg:off)
00000289 FF36[1100]                 push    word [SectorCount]
                                                            ; number of sectors to read
                                    ;mov    bx, 16          ; size of DAP
0000028D B310                       mov     bl, 16
0000028F 53                         push    bx
00000290 89E6                       mov     si, sp
00000292 B442                       mov     ah, 42h
00000294 52                         push    dx
00000295 8A16[3D00]                 mov     dl, [BootDrive]
00000299 1E                         push    ds
0000029A 16                         push    ss
0000029B 1F                         pop     ds

0000029C CD13                       int     13h     ; DISK - IBM/MS Extension - EXTENDED READ
                                                    ; (DL - drive, DS:SI - disk address packet)
0000029E 1F                         pop     ds
0000029F 5A                         pop     dx              ; sector number, hw
000002A0 7209                       jc      short lba_read_err
000002A2 58                         pop     ax              ; size of DAP (disk address packet) = 16
000002A3 58                         pop     ax              ; number of sectors to read
000002A4 50                         push    ax              ; (**) discard ax on stack (StartSecL)
```

```
1613 000002A5 01DC                              add     sp, bx          ; sp points to cx (*)
1614 000002A7 59                                pop     cx      ; (*)   ; remaining retry count value
1615 000002A8 E98400                            jmp     ReadOk
1616                                    lba_read_err:
1617 000002AB 01DC                              add     sp, bx
1618                                            ;;;
1619                                    chs_read:
1620 000002AD 89D0                              mov     ax, dx          ; start sector, hw
1621 000002AF 31D2                              xor     dx, dx ; 0
1622                                            ;;;
1623                                            ; 05/10/2023 - Retro DOS v5.0
1624                                            ; (PCDOS 7.1 IBMBIO.COM - MSLOAD:03E3h
1625 000002B1 3906[2500]                        cmp     [SecPerTrack], ax ; hw of disk (LBA) address
1626 000002B5 7326                              jnb     short DoDivide ; (must not be > sectors per track)
1627                                    ErrorOut:
1628                                    ReadError:
1629                                            ;push   cs
1630                                            ;pop    ds
1631                                            ; ds = cs
1632 000002B7 BE[2904]                          mov     si, NonSystemDiskMsg ; "Non-System disk or disk error" ...
1633                                    WriteTTY:
1634 000002BA AC                                lodsb
1635 000002BB 08C0                              or      al, al
1636 000002BD 7408                              jz      short wait_key_reboot
1637 000002BF B40E                              mov     ah, 0Eh
1638 000002C1 B307                              mov     bl, 7
1639 000002C3 CD10                              int     10h
1640 000002C5 EBF3                              jmp     short WriteTTY
1641                                    wait_key_reboot:
1642 000002C7 30E4                              xor     ah, ah
1643 000002C9 CD16                              int     16h
1644                                            ;xor    bx, bx
1645                                            ;mov    ds, bx ; 0
1646                                            ;les    bx, [cs:OrgDasdPtr]
1647 000002CB C41E[4100]                        les     bx, [OrgDasdPtr] ; Restore ROMBIOS's INT 1Eh vector
1648                                            ; 06/10/2023
1649 000002CF 31F6                              xor     si, si ; 0
1650 000002D1 8EDE                              mov     ds, si
1651
1652 000002D3 BE7800                            mov     si, 78h ; 1Eh*4
1653 000002D6 891C                              mov     [si], bx
1654 000002D8 8C4402                            mov     [si+2], es
1655 000002DB CD19                              int     19h             ; DISK BOOT
1656                                                                    ; causes reboot of disk system
1657                                            ;;;
1658                                    DoDivide:
1659                                            ;;div   word [cs:SecPerTrack]
1660                                            ;div    word [SecPerTrack]
1661                                            ; 24/12/2022
1662 000002DD 8B1E[2500]                        mov     bx, [SecPerTrack]
1663 000002E1 F7F3                              div     bx
1664 000002E3 A3[0B00]                          mov     [TempH], ax
1665                                            ;mov    [cs:TempH], ax
1666 000002E6 58                                pop     ax ; (**) ; 05/10/2023; start sector, lw
1667 000002E7 F7F3                              div     bx
1668                                            ;div    word [SecPerTrack]
1669                                            ;;div   word [cs:SecPerTrack] ; [TempH]:ax = track,
1670                                                                          ; dx = sector number
1671
1672                                            ;mov    bx, [cs:SecPerTrack]  ; Get number of sectors we can
1673                                                                          ; read in this track
1674 000002E9 29D3                              sub     bx, dx          ; dx = start sector on (same) track
1675                                            ;mov    si, bx
1676 000002EB 8B36[1100]                        mov     si, [SectorCount] ; sectors to read on (same) track (remain sectors)
1677
1678 000002EF 39DE                              cmp     si, bx
1679 000002F1 7602                              jna     short GotLength
1680                                            ;cmp    [SectorCount], si
1681                                            ;;cmp   [cs:SectorCount], si  ; Is possible sectors in track more
1682                                            ;jnb    short GotLength               ; than what we need to read?
1683 000002F3 89DE                              mov     si, bx
1684                                            ;mov    si, [SectorCount]
1685                                            ;;mov   si, [cs:SectorCount]  ; Yes, only read what we need to
1686                                    ;GotLength:
1687
1688                                            ; 24/12/2022
1689                                            ; IO.SYS < 40KB (segment override is not possible)
1690                                            ; 700h+0F8FFh < 64KB address
1691                                            ;       (there is not an override risk up to 63743 bytes)
1692                                    ; 24/12/2022
1693                                    %if 0
1694                                            ; 24/12/2022
1695                                            ; dma boundary check for >64KB reads
1696                                            ; Also, Segment Override risk !
1697                                            or      di, di
1698                                            jz      short dma_boundary_ok ; no problem for the 1st read
1699                                            ;cmp    byte [BootDrive], 80h
1700                                            ;;cmp   byte [cs:BootDrive], 80h
1701                                            ;jnb    short dma_boundary_ok ; no problem for hard disks
1702                                    dma_boundary_chk:
1703                                            cmp     si, 1
1704                                            jna     short dma_boundary_ok
1705                                                    ; 1 sector read will not cause a boundary error
1706                                            push    dx
1707                                            push    ax
1708                                            mov     ax, si
1709                                            sub     dx, dx
1710                                            mul     word [BytesPerSec]
1711                                            mov     bx, es
1712                                            mov     cl, 4
1713                                            shl     bx, cl  ; convert paragraphs to bytes
1714                                            ; bx = segment start position (for 64K memory sections)
1715                                            add     bx, ax  ; byte count to read
1716                                            pop     ax
1717                                            pop     dx
1718                                            add     bx, di  ; add current buffer offset to byte count
1719                                            jnc     short dma_boundary_ok
1720
1721                                            ; Sector count must be decreased to prevent
1722                                            ; DMA boundary error or segment override risk!
1723                                            dec     si
1724                                            jmp     short dma_boundary_chk
1725                                    dma_boundary_ok:
1726                                    %endif
1727                                            ; 24/12/2022
1728                                    GotLength:
1729                                            ;inc    dl                      ; Sector numbers are 1-based
1730                                            ; 18/12/2022
1731 000002F5 42                                inc     dx
1732 000002F6 88D3                              mov     bl, dl                  ; Start sector in BL
1733                                            ; 24/12/2022
1734 000002F8 8B16[0B00]                        mov     dx, [TempH]             ; DX:AX = Track
1735                                            ;mov    dx, [cs:TempH]          ; DX:AX = Track
1736 000002FC 50                                push    ax
```

```
1737 000002FD 89D0                      mov     ax, dx
1738 000002FF 31D2                      xor     dx, dx
1739                                    ; 24/12/2022
1740 00000301 F736[2700]                div     word [NumHeads]
1741                                    ;div    word [cs:NumHeads]    ; Start cyl in AX, head in dl
1742                                    ;mov    [TempH], ax
1743                                    ;;mov   [cs:TempH], ax
1744 00000305 58                        pop     ax
1745 00000306 F736[2700]                div     word [NumHeads]
1746                                    ;div    word [cs:NumHeads]    ; [TempH]:AX = Cylinder, DX = Head
1747
1748                                            ; At this moment, we assume that TempH = 0,
1749                                            ; ax <= 1024, dx <= 255
1750
1751 0000030A 88D6                      mov     dh, dl
1752
1753 0000030C B106                      mov     cl, 6
1754 0000030E D2E4                      shl     ah, cl                ; Shift cyl high bits up
1755 00000310 08DC                      or      ah, bl                ; Mix in with sector bits
1756 00000312 88C5                      mov     ch, al                ; Setup cyl low
1757 00000314 88E1                      mov     cl, ah                ; Setup cyl/high - sector
1758 00000316 89FB                      mov     bx, di                ; Get back OFFSET
1759                                    ; 24/12/2022
1760 00000318 8A16[3D00]                mov     dl, [BootDrive]               ; Get drive
1761                                    ;mov    dl, [cs:BootDrive]    ; Get drive
1762 0000031C 89F0                      mov     ax, si                ; Get number of sectors to read (al)
1763 0000031E B402                      mov     ah, 2                 ; Read sectors
1764                                    ; 23/12/2022
1765                                    ;push   ax
1766                                    ;push   di
1767
1768                                            ; Issue one read request. ES:BX have the transfer address,
1769                                            ; AL is the number of sectors.
1770
1771 00000320 CD13                      int     13h             ; DISK - READ SECTORS INTO MEMORY
1772                                            ; AL = number of sectors to read, CH = track, CL = sector
1773                                            ; DH = head, DL       = drive, ES:BX -> buffer to fill
1774                                            ; Return: CF set on error, AH =     status, AL = number of sectors read
1775                                            ; 23/12/2022
1776                                            ;pop    di
1777                                            ;pop    ax
1778
1779                                            ; 09/12/2023
1780                                            ; 23/12/2022
1781                                            ;mov    ah, 0
1782
1783 00000322 59                                pop     cx ; (*)        ; Get retry count back
1784 00000323 730A                              jnc     short ReadOk    ; 23/12/2022
1785
1786                                            ; 23/12/2022
1787                                            ;mov    bx, di          ; Get offset
1788                                            ; 12/12/2023
1789 00000325 30E4                              xor     ah, ah
1790                                            ; ah = 0
1791                                            ; 23/12/2022
1792                                            ;push   cx
1793                                            ; 24/12/2022
1794                                            ;mov    dl, [BootDrive]
1795                                            ;;mov   dl, [cs:BootDrive]
1796                                            ; 23/12/2022
1797                                            ;push   di
1798 00000327 CD13                              int     13h             ; DISK - RESET DISK SYSTEM
1799                                                                    ; DL = drive (if bit 7 is set both hard     disks and floppy disks reset)
1800                                            ; 23/12/2022
1801                                            ;pop    di
1802                                            ;pop    cx
1803 00000329 49                                dec     cx
1804 0000032A 748B                              jz      short ReadError
1805 0000032C E91AFF                            jmp     TryRead
1806                                    ; ---------------------------------------------------------------------------
1807                                    ; 05/10/2023
1808                                    ;ReadError:
1809                                    ;       jmp     ErrorOut
1810                                    ; ---------------------------------------------------------------------------
1811
1812                                    ReadOk:
1813                                            ; 09/12/2023
1814                                            ; al = sectors per cluster or sectors per track
1815                                            ; (al <= 64, sectors per cluster <= 64) ; (!!!)
1816                                            ; ((128*512 = 65536 -> ax=0, dx=1 is unexpected result here!))
1817
1818                                            ; 23/12/2022
1819                                            ; ah = 0
1820                                            ; 22/12/2022
1821                                            ;xor    ah, ah          ; Mask out read command, just get # read
1822                                            ; ch = 0
1823 0000032F 88C1                              mov     cl, al
1824
1825                                            ; 09/12/2023
1826                                            ;; 22/12/2022
1827                                            ;; cx = ax = read (sector) count
1828                                            ;;mov   bx, [cs:BytesPerSec]  ; Bytes per sector
1829                                            ;;mul   bx                    ; Get total bytes read
1830                                            ;; 24/12/2022
1831                                            ;; ds = cs
1832                                            ;mul    word [BytesPerSec]
1833                                            ;;mul   word [cs:BytesPerSec]
1834                                            ; 09/12/2023
1835 00000331 A1[1B00]                         mov     ax, [BytesPerSec]
1836 00000334 F7E1                             mul     cx      ; (!!!)
1837 00000336 01C7                             add     di, ax          ; Add it to OFFSET
1838                                            ; 09/12/2023
1839                                            ; dx = 0 (CL must be <= 64) ; (!!!)
1840
1841                                            ; 24/12/2022
1842                                            ; IO.SYS < 40KB (segment override is not possible)
1843                                            ; 700h+0F8FFh < 64KB address
1844                                            ;       (there is not an override risk up to 63743 bytes)
1845                                            ;add    di, ax
1846                                            ;jnc    short read_next_sector
1847                                            ;mov    bx, es
1848                                            ;;add   bx, 1000h
1849                                            ;add    bh, 10h
1850                                            ;mov    es, bx
1851                                    ;read_next_sector:
1852
1853                                            ; 24/12/2022
1854                                            ; ds = cs
1855                                            ; 22/12/2022
1856 00000338 290E[1100]                       sub     [SectorCount], cx
1857                                            ;sub    [cs:SectorCount], cx
1858                                            ;;sub   [cs:SectorCount], ax  ; Bump number down
1859 0000033C 7464                             jz      short EndRead
1860 0000033E 010E[0700]                       add     [StartSecL], cx
```

```
                                        ;add   [cs:StartSecL], cx
                                        ;;add  [cs:StartSecL], ax    ; Where to start next time
                                        ;adc   word [StartSecH], 0
                                        ; 09/12/2023
00000342 1116[0900]                     adc    [StartSecH], dx ; 0
                                        ;;adc  word [cs:StartSecH], 0
00000346 E9FDFE                         jmp    ReadSectors
                                ; ---------------------------------------------------------------------------

                                        ; 09/12/2023
                                        ; 06/10/2023
                                        ; 24/12/2022
                                ;EndRead:
                                ;       retn

                                ; =============== S U B   R O U T I N E =======================================

                                ; GetNextFatEntry
                                ; ---------------------------------------------------------------------------
                                ;
                                ; NOTES:
                                ;
                                ;   given the last cluster found, this will return the next cluster of
                                ;   iosys. if the last cluster is (f)ff8 - (f)fff, then the final cluster
                                ;   of iosys has been loaded, and control is passed to goto_iosys
                                ;   msload can handle maximum fat area size of 128 kb.
                                ;
                                ; INPUT:
                                ;
                                ;    CS:CurrentCluster
                                ;    CS:FatSize
                                ;
                                ; OUTPUT:
                                ;
                                ;   CS:CurrentCluster (updated)
                                ;
                                ; calls:  GetFatSector
                                ; ---------------------------------------------------------------------------
                                ; get CurrentCluster
                                ;
                                ; if (16 bit fat)
                                ;    {if (CurrentCluster = fff8 - ffff)
                                ;    {jmp goto_iosys}
                                ;     else
                                ;   {get OFFSET by multiply cluster by 2}
                                ;
                                ; else
                                ;    {if (CurrentCluster = ff8 - fff)
                                ;    {jmp goto_iosys}
                                ;     else
                                ;   {get OFFSET by multiply cluster by 3
                                ;
                                ;    rotate right to divide by 2
                                ;
                                ;    if (cy set - means odd number)
                                ;       {shr 4 times to keep high twelve bits}
                                ;
                                ;    else
                                ;       {and with 0fffh to keep low 12 bits}
                                ;   }
                                ;    }
                                ;
                                ; ---------------------------------------------------------------------------

                                ; 09/12/2022
                                ; FAT_12_BIT equ 1
                                ; NOT_END_OF_FILE equ 0  ; ~END_OF_FILE ; END_OF_FILE equ 0FFh

                                GetNextFatEntry:
                                        ; 05/10/2023 - Retro DOS v5.0 IBMBIO.COM (Modified PCDOS 7.1)

00000349 06                             push   es
                                        ; 24/12/2022
                                        ; ds = cs
                                        ;mov   ax, [cs:FatSegment]
0000034A A1[4500]                       mov    ax, [FatSegment]
0000034D 8EC0                           mov    es, ax         ; ES-> FAT area segment
                                        ; 09/12/2022
                                        ;;mov  byte [cs:EndOfFile], END_OF_FILE
                                        ;mov   byte [cs:EndOfFile], 0FFh ; Assume last cluster
                                        ;mov   ax, [cs:CurrentCluster] ; Get last cluster
                                        ; 24/12/2022
                                        ; ds = cs
0000034F C606[4000]FF                   mov    byte [EndOfFile], 0FFh ; Assume last cluster
                                        ;;;
                                        ;mov   ax, [CurrentCluster] ; Get last cluster
                                        ; 05/10/2023 - Retro DOS v5.0 (PCDOS 7.1)
00000354 A1[1300]                       mov    ax, [CurrentClusterL] ; Get last cluster
                                        ; 06/10/2023
                                        ;mov   di, [CurrentClusterH]
                                chk_fat32_type:
                                        ; 05/10/2023
00000357 803E[3E00]0B                   cmp    byte [FatType], 0Bh  ; FAT32 (CHS) fs ?
0000035C 7520                           jne    short chk_fat_type   ; no
                                Got32bit:
0000035E 8B3E[1500]                     mov    di, [CurrentClusterH]
00000362 01C0                           add    ax, ax
00000364 11FF                           adc    di, di
00000366 01C0                           add    ax, ax
00000368 11FF                           adc    di, di          ; Get the FAT offset (di:si)
0000036A 89C6                           mov    si, ax
0000036C E86700                         call   GetFatSector
0000036F 268B07                         mov    ax, [es:bx]
00000372 268B7F02                       mov    di, [es:bx+2]
                                        ; 06/10/2023
                                        ;and   di, 0FFFh ; 28 bit cluster number
00000376 81FFFF0F                       cmp    di, 0FFFh       ; FAT32 cluster numbers are 28 bit numbers
                                                               ; (higher 4 bits are -must be- zero)
                                        ;jne   short GotFAT32ClusterDone
0000037A 751E                           jne    short GotClusterDoneJ
                                        ; 06/10/2023
                                        ;cmp   ax, 0FFF8h
                                ;GotFAT32ClusterDone:
                                        ;jmp   short GotClusterDoneJ
                                        ; 09/12/2023
                                        ;mov   dx, 0FFF8h
                                        ;jmp   short isitlastcluster
0000037C EB17                           jmp    short isitlastcluster_16_32
                                        ;;;
                                chk_fat_type:
                                        ; 06/10/2023
0000037E 29FF                           sub    di, di ; 0 ; hw of the cluster number must be 0
                                                       ; (if it is not FAT32 cluster)
                                        ; 05/10/2023
```

```
1985 00000380 803E[3E00]01                      cmp     byte [FatType], 1
1986                                            ;cmp    byte [Fatsize], 1
1987                                            ;;;cmp  byte [cs:FatSize], FAT_12_BIT
1988                                            ;;cmp   byte [cs:Fatsize], 1
1989                                            ;jne    short Got16Bit ; 23/12/2022
1990                                            ; 09/12/2023
1991 00000385 741C                              je      short Got12Bit
1992
1993                                    ; ---------------------------------------------------------------------------
1994
1995                                            ; 09/12/2023
1996                                    Got16Bit:
1997                                            ; 23/12/2022
1998                                            ;push   dx
1999                                            ;xor    dx, dx
2000                                            ; 05/10/2023
2001                                            ;sub    dx, dx ; 23/12/2022
2002
2003                                            ; 05/10/2023 - Retro DOS v5.0 IBMBIO.COM (MSLOAD)
2004                                            ;shl    ax, 1                   ; Multiply cluster by 2
2005                                            ;adc    dx, 0
2006 00000387 D1E0                              shl     ax, 1
2007 00000389 11FF                              adc     di, di  ; di = 0 ; 06/10/2023
2008
2009 0000038B 89C6                              mov     si, ax                  ; Get the final buffer OFFSET
2010 0000038D E84600                            call    GetFatSector
2011                                            ; 23/12/2022
2012                                            ;pop    dx
2013
2014                                            ; 05/10/2023
2015 00000390 31FF                              xor     di, di          ; HW of cluster number is 0
2016
2017 00000392 268B07                            mov     ax, [es:bx]
2018
2019                                            ; 09/12/2023
2020                                    isitlastcluster_16_32:
2021                                            ; 06/10/2023
2022 00000395 BAF8FF                            mov     dx, 0FFF8h
2023                                            ;jmp    short isitlastcluster
2024                                    ; 09/12/2023
2025                                    ; 06/10/2023
2026                                    %if 1
2027                                    isitlastcluster:
2028 00000398 39D0                              cmp     ax, dx
2029                                            ;cmp    ax, 0FFF8h
2030                                    GotClusterDoneJ:   ; 05/10/2023
2031 0000039A 7305                              jnb     short GotClusterDone
2032                                    NotLastCluster:
2033                                            ; 24/12/2022
2034                                            ; ds = cs
2035                                            ;;mov   byte [cs:EndOfFile], NOT_END_OF_FILE ; ~END_OF_FILE
2036                                            ;mov    byte [cs:EndOfFile], 0; Assume not last cluster
2037 0000039C C606[4000]00                      mov     byte [EndOfFile], 0   ; Assume not last cluster
2038                                    GotClusterDone:
2039 000003A1 07                                pop     es
2040                                            ; 24/12/2022
2041                                    EndRead:
2042 000003A2 C3                                retn
2043                                    %endif
2044
2045                                    ; ---------------------------------------------------------------------------
2046
2047                                            ; 09/12/2023
2048                                    Got12Bit:
2049 000003A3 89C6                              mov     si, ax
2050                                            ;shr    ax, 1
2051                                            ;add    si, ax          ; SI = AX * 1.5 = AX + AX/2
2052                                            ; 05/10/2023
2053                                            ;mov    dx, di
2054                                            ;shr    dx, 1
2055                                            ;rcr    ax, 1
2056                                            ;add    si, ax
2057                                            ;adc    di, dx          ; di:si = dx:ax * 1.5 = dx:ax + dx:ax/2
2058                                            ; 06/10/2023
2059 000003A5 D1E8                              shr     ax, 1
2060 000003A7 01C6                              add     si, ax
2061                                            ;sub    di, di ; 0 ; (di must be 0 for FAT12)
2062
2063                                            ; 23/12/2022
2064                                            ;push   dx
2065                                            ;xor    dx, dx
2066                                            ; 05/10/2023
2067                                            ;sub    dx, dx ; 23/12/2022
2068 000003A9 E82A00                            call    GetFatSector
2069                                            ; 23/12/2022
2070                                            ;pop    dx
2071 000003AC 7510                              jnz     short ClusterOk
2072 000003AE 268A07                            mov     al, [es:bx]
2073                                            ; 22/12/2022
2074                                            ;mov    [cs:TempCluster], al
2075                                            ; 06/10/2023
2076 000003B1 50                                push    ax ; (*)
2077                                            ;inc    si
2078                                            ;add    si, 1
2079                                            ;adc    di, 0
2080                                            ; 05/10/2023
2081 000003B2 46                                inc     si
2082                                            ; 06/10/2023
2083                                            ;jnz    short Got12Bit_rnfs
2084                                            ;inc    di
2085                                    ;Got12Bit_rnfs:
2086                                            ; 23/12/2022
2087                                            ;push   dx
2088                                            ; 05/10/2023
2089                                            ;xor    dx, dx
2090 000003B3 E82000                            call    GetFatSector   ; Read next fat sector
2091                                            ; 23/12/2022
2092                                            ;pop    dx
2093                                            ; 22/12/2022
2094                                            ;mov    al, [es:0]
2095                                            ;mov    [cs:TempCluster+1], al
2096                                            ;mov    ax, [cs:TempCluster]
2097                                            ; 06/10/2023
2098                                            ; 22/12/2022
2099 000003B6 58                                pop     ax ; (*)
2100 000003B7 268A260000                        mov     ah, [es:0]
2101 000003BC EB03                              jmp     short EvenOdd
2102                                    ; ---------------------------------------------------------------------------
2103
2104                                    ClusterOk:
2105 000003BE 268B07                            mov     ax, [es:bx]
2106                                    EvenOdd:
2107                                            ; 24/12/2022
2108                                            ; ds = cs
```

```
                                        ;test   byte [CurrentCluster], 1
                                        ; 05/10/2023 - Retro DOS v5.0 IBMBIO.COM (MSLOAD)
000003C1 F606[1300]01                   test    byte [CurrentClusterL], 1
                                        ; 10/12/2022
                                        ;test   byte [cs:CurrentCluster], 1 ; 09/12/2022
                                        ;;test  word [cs:CurrentCluster], 1 ; Was last cluster odd?
000003C6 7505                           jnz     short OddResult             ; If not zero it was odd
000003C8 25FF0F                         and     ax, 0FFFh           ; Keep low 12 bits
000003CB EB04                           jmp     short TestEOF
                                ; ---------------------------------------------------------------------------

                                OddResult:
000003CD B104                           mov     cl, 4               ; Keep high 12 bits for odd
000003CF D3E8                           shr     ax, cl
                                TestEOF:
                                        ; 06/10/2023
                                        ; di = 0
                                        ;xor    di, di          ; HW of cluster number is 0

                                        ; 06/10/2023
                                        ;cmp    ax, 0FF8h           ; Is it last cluster?
                                        ;jnb    short GotClusterDone  ; Yep, all done here
                                        ;jmp    short NotLastCluster
000003D1 BAF80F                         mov     dx, 0FF8h
                                        ; 09/12/2023
000003D4 EBC2                           jmp     short isitlastcluster
                                ; 09/12/2023
                                ; 06/10/2023
                                %if 0
                                isitlastcluster:
                                        ; 06/10/2023
                                        cmp     ax, dx
                                GotClusterDoneJ:
                                        jnb     short GotClusterDone
                                NotLastCluster:
                                        ; 24/12/2022
                                        ; ds = cs
                                        ;;mov   byte [cs:EndOfFile], NOT_END_OF_FILE ; ~END_OF_FILE
                                        ;mov    byte [cs:EndOfFile], 0; Assume not last cluster
                                        mov     byte [EndOfFile], 0   ; Assume not last cluster
                                GotClusterDone:
                                        pop     es
                                        ; 06/10/2023
                                        ; 24/12/2022
                                ;EndRead:
                                        retn
                                %endif

                                ; GetFatSector
                                ; ---------------------------------------------------------------------------
                                ;function: find and read the corresponding fat sector into ES:0
                                ;
                                ;in). SI = offset value (starting from fat entry 0) of fat entry to find.
                                ;     ES = fat sector segment
                                ;     CS:BytesPerSec
                                ;
                                ;out). corresponding fat sector read in.
                                ;      BX = offset value of the corresponding fat entry in the fat sector.
                                ;      CX destroyed.
                                ;      zero flag set if the fat entry is splitted, i.e. when 12 bit fat entry
                                ;      starts at the last byte of the fat sector. in this case, the caller
                                ;      should save this byte, and read the next fat sector to get the rest
                                ;      of the fat entry value. (this will only happen with the 12 bit fat).
                                ;
                                ; ---------------------------------------------------------------------------

                                ; 09/12/2023
                                ; 06/10/2023 - Retro DOS v5.0 IBMBIO.COM (Modified PCDOS 7.1)
                                ; (PCDOS 7.1 IBMBIO.COM - MSLOAD:054Ch)

                                        ; 24/12/2022
                                        ; 22/12/2022
                                GetFatSector:
                                        ;push   ax      ; 06/10/2023
000003D6 56                             push    si
000003D7 57                             push    di              ; di:si = byte offset in (entire) FAT
                                        ;;;
                                        ; 06/10/2023
000003D8 89F8                           mov     ax, di          ; 32 bit division (dx:ax/512)
000003DA 8B0E[1B00]                     mov     cx, [BytesPerSec]
000003DE 31D2                           xor     dx, dx
000003E0 F7F1                           div     cx
000003E2 89C3                           mov     bx, ax
                                        ;;;
000003E4 89F0                           mov     ax, si
                                        ; 24/12/2022
                                        ; ds = cs
                                        ;mov    cx, [cs:BytesPerSec]
                                        ; 06/10/2023
000003E6 F7F1                           div     cx                      ; AX = Sector number, DX = Offset
                                        ;div    word [BytesPerSec]
                                                ; dx = byte offset in the FAT sector
                                                ; ax = low word of the FAT sector number
                                                ; bx = high word of the FAT sector number
                                        ; 06/10/2023
000003E8 3B1E[0F00]                     cmp     bx, [LastFatSectorH]  ; FAT32 (32 bit cluster numbers)
000003EC 7506                           jne     short not_same_fat_sector
000003EE 3B06[0D00]                     cmp     ax, [LastFatSectorL]
                                        ;cmp    ax, [LastFatSector]
                                        ;;cmp   ax, [cs:LastFatSector]; The same fat sector?
000003F2 742D                           je      short SplitChk        ; Don't need to read it again.
                                not_same_fat_sector:
                                        ; 06/10/2023
000003F4 A3[0D00]                       mov     [LastFatSectorL], ax
000003F7 891E[0F00]                     mov     [LastFatSectorH], bx
                                        ;
                                        ;mov    [LastFatSector], ax
                                        ;;mov   [cs:LastFatSector], ax
                                        ; 06/10/2023
000003FB 51                             push    cx ; ** ; bytes per sector
                                        ;
000003FC 52                             push    dx ; *
                                        ; 24/12/2022
                                        ;xor    dx, dx
                                        ; 06/10/2023
000003FD 89DA                           mov     dx, bx
                                        ;
                                        ;add    ax, [cs:HiddenSectorsL]
                                        ;adc    dx, [cs:HiddenSectorsH]
                                        ;add    ax, [cs:ReservSectors]
                                        ;adc    dx, 0
                                        ; 24/12/2022
                                        ; ds = cs
                                        ; 06/10/2023 - Retro DOS v5.0 (PCDOS 7.1 IBVBIO.COM)
```

```
                                        ;add    ax, [FatStartSecL]
                                        ;adc    dx, [FatStartSecH]
                                        ; 09/12/2023
000003FF 31FF                           xor     di, di ; 0
00000401 0306[2900]                     add     ax, [HiddenSectorsL]
00000405 1316[2B00]                     adc     dx, [HiddenSectorsH]
00000409 0306[1F00]                     add     ax, [ReservSectors]
                                        ;adc    dx, 0
                                        ; 09/12/2023
0000040D 11FA                           adc     dx, di ; 0

0000040F A3[0700]                       mov     [StartSecL], ax
00000412 8916[0900]                     mov     [StartSecH], dx             ; Set up for ReadSectors
                                        ;mov    [cs:StartSecL], ax
                                        ;mov    [cs:StartSecH], dx    ; Set up for ReadSectors

00000416 C706[1100]0100                 mov     word [SectorCount], 1 ; 1 sector
                                        ;mov    word [cs:SectorCount], 1 ; 1 sector
                                        ; 06/10/2023
                                        ; di = 0
                                        ;xor    di, di ; 0
                                        ; es:di = FATSEGMENT:0000h
0000041C E827FE                         call    ReadSectors
0000041F 5A                             pop     dx ; *
                                        ; 06/10/2023
                                        ;mov    cx, [BytesPerSec]
00000420 59                             pop     cx ; **
                                        ; 24/12/2022
                                        ;mov    cx, [cs:BytesPerSec]
                                SplitChk:
                                        ; 06/10/2023
                                        ; cx = bytes per sector
                                        ; 24/12/2022
                                        ;mov    cx, [BytesPerSec]
00000421 49                             dec     cx                  ; CX = SECTOR SIZE - 1
00000422 39CA                           cmp     dx, cx              ; If last byte of sector, splitted entry.
00000424 89D3                           mov     bx, dx              ; set bx to dx
00000426 5F                             pop     di
00000427 5E                             pop     si
                                        ;pop    ax      ; 06/10/2023
                                EndWrite:               ; 10/12/2022
00000428 C3                             retn

                                ; ---------------------------------------------------------------------------

                                ; 05/10/2023 - Retro DOS v5.0 IBMBIO.COM (Modified PCDOS 7.1)
                                %if 0

                                ErrorOut:
                                        ; 24/12/2022
                                        ; ds = cs
                                        ;push   cs
                                        ;pop    ds

                                        mov     si, NonSystemDiskMsg ; "\r\nNon-System disk or disk error\r\nRe"...
                                        call    WriteTTY

                                        ; Wait for a keypress on the keyboard.
                                        ; Use the bios keyboard interrupt.

                                        xor     ah, ah
                                        int     16h             ; KEYBOARD - READ CHAR FROM BUFFER, WAIT IF EMPTY
                                                                ; Return: AH = scan code, AL = character

                                        ; We have to restore the address of the original rom disk
                                        ; parameter table to the location at [0:DskAddr]. The address
                                        ; of this original table has been saved previously in
                                        ; 0:OrgDasdPtr and 0:OrgDasdPtr+2. After this table address
                                        ; has been restored we can reboot by invoking the bootstrap
                                        ; loader bios interrupt.

                                        ; 23/12/2022
                                        ;xor    bx, bx
                                        ;mov    ds, bx
                                        ;les    bx, [OrgDasdPtr] ; Wrong DS segment !
                                                                 ; (Erdogan Tan, 23/12/2022)
                                        les     bx, [OrgDasdPtr] ; Correct DS segment = CS

                                        ; 23/12/2022
                                        push    ss ; 0
                                        pop     ds
                                        ; ds = 0

                                        mov     si, DskAddr     ; (Int 1Eh)
                                        mov     [si], bx        ; restore offset
                                        mov     [si+2], es      ; restore segment
                                        int     19h             ; reboot

                                ; =============== S U B   R O U T I N E =======================================

                                ; WriteTTY
                                ; ---------------------------------------------------------------------------
                                ; in) DS:si -> asciiz string.
                                ;
                                ; WriteTTY the character in al to the screen.
                                ; use video service 'write teletype to active page' (ROM_TTY)
                                ; use normal character attribute
                                ; ---------------------------------------------------------------------------

                                WriteTTY:
                                        lodsb
                                        or      al, al
                                        jz      short EndWrite
                                        ;mov    AH, ROM_TTY     ; 09/12/2022
                                        mov     ah, 0Eh
                                        mov     bl, 7   ; "normal" attribute
                                        int     10h     ; - VIDEO - WRITE CHARACTER AND      ADVANCE CURSOR (TTY WRITE)
                                                        ; AL = character, BH = display page (alpha modes)
                                                        ; BL = foreground color       (graphics modes)
                                        jmp     short WriteTTY
                                ; ---------------------------------------------------------------------------

                                ; 10/12/2022
                                ;EndWrite:
                                ;       retn

                                %endif      ; 05/10/2023

                                ; ---------------------------------------------------------------------------

                                ; 06/10/2023 - Retro DOS v5.0 IBMBIO.COM (IO.SYS) ((Modified PCDOS 7.1))
                                ; (PCDOS 7.1 IBMBIO.COM - MSLOAD:054Ch)

                                ; 09/12/2022
```

```
                                ;include msbio.cl1
                                
                                        ; 22/12/2022
                                        ; 20/12/2022
                                        ; 18/12/2022
                                        ;db 0   ; (word alignment)
                                NonSystemDiskMsg:
00000429 0D0A                           db 0Dh,0Ah              ; ...
0000042B 4E6F6E2D5379737465-            db 'Non-System disk or disk error',0Dh,0Ah
00000434 6D206469736B206F72-
0000043D 206469736B20657272-
00000446 6F720D0A
0000044A 5265706C6163652061-            db 'Replace and press any key when ready',0Dh,0Ah,0
00000453 6E6420707265737320-
0000045C 616E79206B65792077-
00000465 68656E207265616479-
0000046E 0D0A00
                                ; 25/12/2022
00000471 90<rep Fh>             align 16
                                
                                EndOfLoader:        ; (PCDOS 7.1 IBMBIO.COM - MSLOAD:05F0h) ; 06/10/2023
                                        ;dw 01A1h       ; 10/12/2022
                                
                                ; ----------------------------------------------------------------------------
                                
                                ;============================================================================
                                ; DOS BIOS (IO.SYS) DATA SEGMENT
                                ;============================================================================
                                ; 09/12/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ; 30/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; 02/10/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
                                
                                section .BIOSDATA vstart=0
                                
                                ;--- DOSBIOS data segment ---------------------------------------------------
                                ;----------------------------------------------------------------------------
                                
                                ;Bios_Data segment
                                
                                BData_start:
00000000 E9931B                 hdrv_pat:   jmp     init            ; MSBIO1.ASM, MSSBDATA.INC
                                ; ---------------------------------------------------------------------------
                                
00000003 0000                   DosDataSg:  dw 0
                                
                                ; DOS's int 2f handler will exit via a jump through here.
                                ; This is how the BIOS hooks int2f
                                
                                ;BIOSDATA:0005h: ; 10/05/2023 (Note the 'bios_i2f equ 5' in 'msdos6.s')
                                
00000005 EA                     bios_i2f:   db 0EAh                 ; far jump to int_2f (segment may not be at 70h)
                                            ; 07/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                            ; PCDOS 7.1 IBMBIO.COM - BIODATA:0006h
                                            ;dw int_2f
                                            ;dw 70h                 ; BIOSDATA segment (KERNEL_SEGMENT)
00000006 [2B16]                             dw i2f_handler
                                bios_i2f_seg:       ; 10/08/2023
00000008 2D03                               dw DOSBIOCODESEG        ; 02CCh for MSDOS 6.21 IO.SYS (25Ch+070h)
                                                                    ; 0364h PCDOS 7.1 IBMBIO.COM  (2F4h+070h)
                                
0000000A 0000                   romstartaddr:       dw 0                    ; The start address for the romfind routines
                                                                    ; This is to maintain binary compatibility
                                                                    ; with DISK based DOS 5.0
                                
                                ; This is a byte used for special key handling in the resident
                                ; console device driver. It must be here so that it can be included
                                ; in the WIN386 instance table (in INC\LMSTUB.ASM).
                                
0000000C 00                     altah:              db 0                    ; special key handling
                                
0000000D 00                     inHMA:              db 0                    ; flag indicates we're running from HMA
0000000E 00000000               xms:        dd 0                    ; entry point to xms if above is true
                                
                                ; PTRSAV - pointer save
                                ;
                                ; This variable holds the pointer to the Request Header passed by a program
                                ; wishing to use a device driver. When the strategy routine is called it
                                ; puts the address of the Request header in this variable and returns.
                                
00000012 00000000               ptrsav:             dd 0
                                auxbuf:             ;db 4 dup(0)            ; set of 1 byte buffers for com 1,2,3, and 4
00000016 00000000                           db 0, 0, 0, 0 ; 19/10/2022
0000001A 0000                   zeroseg:    dw 0                    ; easy way to load segment registers with zero
0000001C 0000                   i13_ds:             dw 0                    ; ds register for int13 call through
0000001E 0000                   prevoper:   dw 0                    ; holds int 13 request (i.e. register ax).
00000020 00                     number_of_sec:      db 0                    ; holds number of secs. to read on an ecc error
00000021 0000                   auxnum:             dw 0                    ; which aux device was requested
                                
                                ;----------------------------------------------------------------------------
                                
                                res_dev_list:
                                
                                ; Device Header for the CON Device Driver
                                
                                CONHeader:                          ; HEADER FOR DEVICE "CON"
00000023 [3500]                             dw auxdev2
00000025 7000                               dw 70h
00000027 1380                   word_727:   dw 8013h
00000029 [1506]                             dw strategy
0000002B [2006]                             dw con_entry
0000002D 434F4E2020202020      aCon:                db 'CON     '
00000035 [4700]                 auxdev2:    dw prndev2              ; HEADER FOR DEVICE "AUX"
00000037 7000                               dw 70h
00000039 0080                               dw 8000h
0000003B [1506]                             dw strategy
0000003D [4106]                             dw aux0_entry
0000003F 4155582020202020      aAux:                db 'AUX     '
00000047 [5900]                 prndev2:    dw timdev               ; HEADER FOR DEVICE "PRN"
00000049 7000                               dw 70h
0000004B C0A0                   word_74B:   dw 0A0C0h
0000004D [1506]                             dw strategy
0000004F [2506]                             dw prn0_entry
00000051 50524E2020202020      aPrn:                db 'PRN     '          ; HEADER FOR DEVICE "CLOCK$"
00000059 [6B00]                 timdev:              dw dskdev
0000005B 7000                               dw 70h
0000005D 0880                               dw 8008h
0000005F [1506]                             dw strategy
00000061 [5906]                             dw tim_entry
00000063 434C4F434B242020      aClock:              db 'CLOCK$  '
0000006B [7B00]                 dskdev:              dw com1dev              ; HEADER FOR DISK DEVICES
0000006D 7000                               dw 70h
                                            ;dw 8C2h
                                            ; 02/10/2023 - Retro DOS v5.0
0000006F C248                               dw 48C2h                ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:006Fh
```

```
                                        ;dw offset strategy
                                        ;dw offset dsk_entry
                                        ; 19/10/2022
00000071 [1506]                         dw strategy
00000073 [5E06]                         dw dsk_entry

                                ; maximum number of drives

00000075 04                     drvmax:         db 4
00000076 FE                     step_drv:   db 0FEh ; -2        ; last drive accessed
00000077 00                     fhave96:    db 0                ; flag to indicate presence of
                                                                ; 96tpi support
00000078 00                     single:         db 0                    ; used to detect single drive systems
00000079 00                     fhavek09:   db 0                ; indicates if this is a k09 or not
                                                                ; used by console driver.
0000007A 00                     fsetowner:  db 0                ; = 1 if we are setting the owner of a
                                                                ; drive. (examined by checksingle).

0000007B [8D00]                 com1dev:    dw lpt1dev          ; Device Header for device "COM1"
0000007D 7000                               dw 70h
0000007F 0080                               dw 8000h
00000081 [1506]                             dw strategy
00000083 [4106]                             dw aux0_entry
00000085 434F4D3120202020       aCom1:              db 'COM1    '
0000008D [9F00]                 lpt1dev:    dw lpt2dev          ; Device Header for device LPT1
0000008F 7000                               dw 70h
00000091 C0A0                               dw 0A0C0h
00000093 [1506]                             dw strategy
00000095 [2C06]                             dw prn1_entry
00000097 4C50543120202020       aLpt1:              db 'LPT1    '
0000009F [B800]                 lpt2dev:    dw lpt3dev          ; Device Header for device LPT2
000000A1 7000                               dw 70h
000000A3 C0A0                               dw 0A0C0h
000000A5 [1506]                             dw strategy
000000A7 [3306]                             dw prn2_entry
000000A9 4C5054322020202000-    aLpt2:              db 'LPT2    ',0,0,0
000000B2 0000

                                ;M058; Start of changes
                                ; Orig13 needs to be at offset 0B4h for the CMS floppy driver to work.
                                ;These guys patch Orig13 with their own int 13h hook and so this offset
                                ;cannot change for them to work. Even ProComm does this.

000000B4 00000000               Orig13:         dd 0                    ; to make Orig13 offset 0B4h

000000B8 [CA00]                 lpt3dev:    dw com2dev          ; Device Header for device LPT3
000000BA 7000                               dw 70h
000000BC C0A0                               dw 0A0C0h
000000BE [1506]                             dw strategy
000000C0 [3A06]                             dw prn3_entry
000000C2 4C50543320202020       aLpt3:              db 'LPT3    '
000000CA [DC00]                 com2dev:    dw com3dev          ; Device Header for device "COM2"
000000CC 7000                               dw 70h
000000CE 0080                               dw 8000h
000000D0 [1506]                             dw strategy
000000D2 [4706]                             dw aux1_entry
                                            ; 19/10/2022
000000D4 434F4D3220202020       aCom2:              db 'COM2    '
                                com3dev:    ;dw offset com4dev  ; Device Header for device "COM3"
000000DC [EE00]                             dw com4dev
000000DE 7000                               dw 70h
000000E0 0080                               dw 8000h
                                            ;dw offset strategy
                                            ;dw offset aux2_entry
000000E2 [1506]                             dw strategy
000000E4 [4D06]                             dw aux2_entry
000000E6 434F4D3320202020       aCom3:              db 'COM3    '
000000EE FFFF                   com4dev:    dw 0FFFFh           ; Device Header for device "COM4"
000000F0 7000                               dw 70h
000000F2 0080                               dw 8000h
000000F4 [1506]                             dw strategy
000000F6 [5306]                             dw aux3_entry
000000F8 434F4D3420202020                   db 'COM4    '

                                ;-----------------------------------------------------------------------------

00000100 10                     RomVectors: db 10h
00000101 00000000               Old10:              dd 0
00000105 13                                 db 13h
00000106 00000000               Old13:              dd 0
0000010A 15                                 db 15h
0000010B 00000000               Old15:              dd 0
0000010F 19                                 db 19h
00000110 00000000               Old19:              dd 0
00000114 1B                                 db 1Bh
00000115 00000000               Old1B:              dd 0

                                ;EndRomVectors      equ $

                                ;NUMROMVECTORS      equ ((EndRomVectors - RomVectors)/5)

                                ;-----------------------------------------------------------------------------

00000119 [5203]                 start_bds:  dw bds1             ; Start of linked list of BDS's
0000011B 7000                               dw 70h              ; KERNEL_SEGMENT

                                ; (MSDOS 3.3) NOTE:
                                ; Some floppy drives do not have changeline support. The result is a
                                ; large amount of inefficiency in the code. A media-check always returns
                                ; "I don't know". This cause DOS to reread the FAT on every access and
                                ; always discard any cached data.
                                ;   We get around this inefficiency by implementing a "Logical Door Latch".
                                ; The following three items are used to do this. The logical door latch is
                                ; based on the premise that it is not physically possible to change floppy
                                ; disks in a drive in under two seconds (most people take about 10). The
                                ; logical door latch is implemented by saving the time of the last successful
                                ; disk operation (in the value TIM_DRV). When a new request is made the
                                ; current time is compared to the saved time. If less than two seconds have
                                ; passed then the value "No Change" is returned. If more than two seconds
                                ; have passed the value "Don't Know" is returned.
                                ;   There is one complecation to this algorithm. Some programs change the
                                ; value of the timer. In this unfortunate case we have an invalid timer.
                                ; This possibility is detected by counting the number of disk operations
                                ; which occur without any time passing. If this count exceeds the value of
                                ; "AccessMax" we assume the counter is invalid and always return "Don't
                                ; Know". The variable "AccessCount" is used to keep track of the number
                                ; of disk operation which occur without the time changing.

0000011D 00                     accesscount:        db 0
0000011E FF                     tim_drv:    db 0FFh
0000011F 00                     medbyt:             db 0
                                wrtverify:  ; 15/10/2022
00000120 02                     rflag:              db 2                ; 2 for read, 3 for write
00000121 00                     verify:             db 0                ; 1 if verify after write
```

```
                        seccnt:         dw 0
                                db 0                    ; -- pad where hardnum was
                        dsktnum:    db 1                ; number of diskette drives

                        ; (MSDOS 3.3) NOTE:
                        ; Some of the older versions of the IBM rom-bios always assumed a seek would
                        ; have to be made to read the diskette. Consequently a large head settle
                        ; time was always used in the I/O operations. To get around this problem
                        ; we need to continually adjust the head settle time. The following
                        ; algorithm is used:
                        ;
                        ;   Get the current head settle value.
                        ;   If it is 1, then
                        ;   set slow = 15
                        ;   else
                        ;   set slow = value
                        ;   ...
                        ;   if we are seeking and writing then
                        ;   use slow
                        ;   else
                        ;   use fast
                        ;   ...
                        ;   restore current head settle value

                        motorstartup:       db 0                ; value from table
                        settlecurrent:      db 0                ; value from table
                        settleslow: db 0                ; slow settle value
                        nextspeed:  db 0                ; value of speed to be used
                        save_head_sttl:     db 0                ; used by read_sector routine
                        save_eot:   db 0                ; saved eot from the default DPT
                        eot:        db 9
                        dpt:        dd 0                ; pointer to Disk Parameter Table
                        cursec:         db 0                ; current sector
                        curhd:          db 0                ; current head
                        curtrk:         dw 0                ; current track
                        spsav:          dw 0                ; save the stack pointer
                        formt_eot:  db 8                ; eot used for format
                        hdnum:          db 0                ; head number
                        trknum:         dw 0                ; track being manipulated
                        gap_patch:  db 50h              ; format gap patched into dpt

                        ;--------------------------------------------------------------------------

                        ; disk errors returned from the IBM rom

                        errin:          db 0CCh             ; write fault (hard disk)
                                db 80h                  ; write fault (hard disk)
                                db 40h                  ; seek failed
                                db 10h                  ; uncorrectable CRC or ECC error on read
                                db 8                    ; dma overrun
                                db 6                    ; disk changed (floppy)
                                db 4                    ; sector not found/read error
                                db 3                    ; disk write-protected
                                ; 02/10/2023 - Retro DOS 5.0 (PCDOS 7.1 IBMBIO.COM)
                                db 1                    ; invalid function in AH or invalid parameter
                                db 0B2h                 ; volume not removable
                                ;
                        lsterr:         db 0                ; all other errors

                        ; returned error codes corresponding to above

                        errout:         db 10               ; write fault error
                                db 2                    ; no response (timeout)
                                db 6                    ; seek failure
                                db 4                    ; bad crc
                                db 4                    ; dma overrun
                                db 15                   ; invalid media       change
                                db 8                    ; sector not found
                                db 0                    ; write attempt to write-protect disk
                                ; 02/10/2023
                                db 3                    ; unknown command error
                                db 3                    ; unknown command error
                                ;
                                db 12                   ; general error

                        ;--------------------------------------------------------------------------
                        ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:0152h

                        ; 02/10/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
                        %if 1
                        disksector: ;times 174 db 0
                        NUM174 equ 512-$
                                times NUM174 db 0
                        JB_sign     :       ;dw 424Ah           ; 'BJ' (nasm) ; 'JB' (masm)
                                dec     dx
                                inc     dx
                                jmp     BData_start     ; db 0E9h, 0FBh, 0FDh

                        IBMBIOCOM$: db '@#IBM:12.01.2003.build_1.32#@ IBMBIO.COM(USA)',0

                                ;times 287 db 0
                                times (disksector+512-$) db 0  ; 287
                        %endif

                        ;--------------------------------------------------------------------------

                        ; 30/12/2018 - Retro DOS v4.0

                        ; read in boot sector here, read done in readboot.
                        ; also read sector for dma check for hard disk.
                        ;
                        ; This buffer is word aligned because certain AMI BIOSs have a bug
                        ; in them which causes the byte after the buffer to be trashed
                        ; on floppy reads to odd-byte boundaries. Although no general effort
                        ; is made to enforce this in the bigger picture, this one small sacrifice
                        ; makes that system more-or-less work.

                        ; 02/10/2023
                        %if 0

                        disksector: ;db 512 dup(0)          ; read in boot sector here
                                ; 19/10/2022
                                times 512 db 0
                        %endif

                        ;--------------------------------------------------------------------------

                        ; 02/10/2023 - Retro DOS v5.0
                        ; 30/12/2018 - Retro DOS v4.0
```

```
2716                                    ;--------------------------------------------------------------------------------
2717                                    ; 25/05/2018 (04/04/2018)
2718                                    ;********************************************************************************
2719                                    ;   "bds" contains information for each drive in the system.
2720                                    ;   various values are patched whenever actions are performed.
2721                                    ;   sectors/alloc. unit in bpb initially set to -1 to signify that
2722                                    ;   the bpb has not been filled. link also set to -1 to signify end
2723                                    ;   of list. # of cylinders in maxparms initialized to -1 to indicate
2724                                    ;   that the parameters have not been set.
2725                                    
2726                                    bds1:               ;dw offset bds2
2727 00000352 [E803]                                  dw bds2 ; 19/10/2022
2728 00000354 7000                                  dw 70h                  ; dword link to next structure
2729 00000356 00                                    db 0                    ; int 13h drive       number
2730 00000357 00                                    db 0                    ; logical drive       letter
2731 00000358 0002                  fdrive1:        dw 512
2732                                                                        ; physical sector size in bytes
2733 0000035A FF                                    db 0FFh                 ; sectors/allocation unit
2734 0000035B 0100                                  dw 1                    ; reserved sectors for dos
2735 0000035D 02                                    db 2                    ; no of file allocation       tables
2736 0000035E 4000                                  dw 64                   ; number of root directory entries
2737 00000360 6801                                  dw 360                  ; number sectors (at 512 bytes each)
2738 00000362 00                                    db 0                    ; media descriptor, initially 0
2739 00000363 0200                                  dw 2                    ; number of fat       sectors
2740 00000365 0900                                  dw 9                    ; sector limit (sectors       per track)
2741 00000367 0100                                  dw 1                    ; head limit (number of       heads - 1)
2742                                                ;
2743                                                ; 02/10/2023
2744                                                ; MSDOS 5.0-6.22 (& PCDOS 7.0)
2745                                                ;dw 0                   ; hidden sector       count (low word)
2746                                                ;dw 0                   ; hidden sector       (high)
2747                                                ;dw 0                   ; number sectors (low)
2748                                                ;dw 0                   ; number sectors (high)
2749                                                ;db 0                   ; true => large       fats
2750                                                ; 02/10/2023
2751                                                ; PCDOS 7.1 (FAT32 support)
2752 00000369 00000000                             dd 0                    ; hidden sector count
2753 0000036D 00000000                             dd 0                    ; number of sectors (32 bit)
2754 00000371 00000000                             dd 0                    ; BPB_FATSz32 ; FAT32 FAT size in sectors ; 4 bytes
2755                                                                        ;   BS_DrvNum ; FAT INT 13h drive number ; 1 byte
2756                                                                        ;   BS_Reserved1 ; FAT reserved byte = 0 ; 1 byte
2757                                                                        ;   BS_BootSig ; FAT Extended boot signature = 29h ; 1 byte
2758                                                                        ;   BS_VolID ; FAT Volume serial number ; 4 bytes
2759 00000375 0000                                 dw 0                    ; BPB_ExtFlags ; FAT32 Extended Flags
2760 00000377 0000                                 dw 0                    ; BPB_FSVer ; FAT32 fs/volume version
2761 00000379 00000000                             dd 0                    ; BPB_RootClus ; FAT32 root directory's first cluster number
2762 0000037D FFFF                                 dw 0FFFFh               ; BPB_FSInfo ; FAT32 FSINFO sector number = -1 (initial)
2763 0000037F FFFF                                 dw 0FFFFh               ; BPB_BkBootSec ; FAT32 backup boot sector number = -1 (initial)
2764 00000381 00<rep Ch>                           times 12 db 0           ; BPB_Reserved  ; FAT32 reserved field = 0, 12 bytes
2765 0000038D 00                                   db 0                    ; true => large       fats
2766                                                ;
2767 0000038E 0000                                 dw 0                    ; open ref. count
2768 00000390 03                                   db 3                    ; form factor
2769 00000391 2000                                 dw 20h                  ; various flags
2770 00000393 2800                                 dw 40                   ; number of cylinders
2771 00000395 0002                  recommended_bps: dw 512                     ; recommended bps for this drive
2772 00000397 01                                   db 1
2773 00000398 0100                                 dw 1
2774 0000039A 02                                   db 2
2775 0000039B E000                                 dw 224                  ; number of root directory entries
2776 0000039D 6801                                 dw 360
2777 0000039F F0                                   db 0F0h                 ; media descriptor, initially 0F0h
2778 000003A0 0200                                 dw 2
2779 000003A2 0900                                 dw 9
2780 000003A4 0200                                 dw 2
2781                                                ;
2782                                                ; 02/10/2023
2783                                                ;dw 0
2784                                                ;dw 0
2785                                                ;dw 0
2786                                                ;dw 0
2787                                                ;;db 6 dup(0)
2788                                                ;times 6 db 0           ; 19/10/2022
2789 000003A6 00000000                             dd 0
2790 000003AA 00000000                             dd 0
2791 000003AE 00000000                             dd 0
2792 000003B2 0000                                 dw 0
2793 000003B4 0000                                 dw 0
2794 000003B6 00000000                             dd 0
2795 000003BA FFFF                                 dw 0FFFFh
2796 000003BC FFFF                                 dw 0FFFFh
2797                                                ;db 12 dup(0)
2798 000003BE 00<rep Ch>                           times 12 db 0           ; 02/10/2023
2799                                                ;
2800 000003CA FF                                   db 0FFh                 ; last track accessed on this drive
2801 000003CB FFFF                                 dw 0FFFFh               ; keep these two contiguous (?)
2802 000003CD FFFF                                 dw 0FFFFh
2803 000003CF 4E4F204E414D452020-                  db 'NO NAME    ',0      ; volume id for this disk
2803 000003D8 202000
2804 000003DB 00000000                             dd 0                    ; current volume serial       from boot record
2805 000003DF 464154313220202000                   db 'FAT12   ',0         ; current file system id from boot record
2806                                    ; ----
2807
2808                                    ; 02/10/2023
2809                                    ; PCDOS 7.1
2810                                    %if 1
2811
2812                                    bds2:               ; 02/10/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
2813 000003E8 FFFF                                 dw 0FFFFh ; -1
2814 000003EA 7000                                 dw 70h
2815 000003EC 00                                   db 0
2816 000003ED 00                                   db 0
2817 000003EE 0002                  fdrive2:       dw 512
2818 000003F0 FF                                   db 0FFh
2819 000003F1 0100                                 dw 1
2820 000003F3 02                                   db 2
2821 000003F4 4000                                 dw 64
2822 000003F6 6801                                 dw 360
2823 000003F8 00                                   db 0
2824 000003F9 0200                                 dw 2
2825 000003FB 0900                                 dw 9
2826 000003FD 0100                                 dw 1
2827 000003FF 00000000<rep 5h>                     times 5 dd 0
2828 00000413 FFFFFFFF                             dd 0FFFFFFFFh
2829 00000417 00000000<rep 3h>                     times 3 dd 0
2830 00000423 00                                   db 0
2831 00000424 0000                                 dw 0
2832 00000426 03                                   db 3
2833 00000427 2000                                 dw 20h
2834 00000429 2800                                 dw 40
2835                                    recbpb2:
2836 0000042B 0002                                 dw 512
2837 0000042D 01                                       db 1
2838 0000042E 0100                                     dw 1
```

```
2839 00000430 02                                    db 2
2840 00000431 E000                                  dw 224
2841 00000433 6801                                  dw 360
2842 00000435 F0                                    db 0F0h
2843 00000436 0200                                  dw 2
2844 00000438 0900                                  dw 9
2845 0000043A 0200                                  dw 2
2846 0000043C 00000000<rep 5h>                  times 5 dd 0
2847 00000450 FFFFFFFF                          dd 0FFFFFFFFh
2848 00000454 00000000<rep 3h>                      times 3 dd 0
2849 00000460 FF                                    db 0FFh
2850 00000461 FFFFFFFF                              dd 0FFFFFFFFh
2851 00000465 4E4F204E414D452020-                   db 'NO NAME    ',0
2851 0000046E 202000
2852 00000471 00000000                              dd 0
2853 00000475 464154313220202000                    db 'FAT12   ',0
2854                                    %endif
2855
2856                                    ; ----
2857
2858                                    ; 02/10/2023
2859                                    ; MSDOS 5.0 - 6.22 (& PCDOS 7.0)
2860                                    %if 0
2861
2862                                    bds2:               dw bds3
2863                                                dw 70h
2864                                                db 0
2865                                                db 0
2866                                    fdrive2:    dw 512
2867                                                db 0FFh, 1, 0, 2, 40h, 0, 68h, 1, 0, 2,     0, 9, 0, 1, 0
2868                                                db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 3, 20h, 0, 28h, 0
2869                                                db 0, 2, 1, 1, 0, 2, 0E0h, 0, 68h, 1, 0F0h, 2, 0, 9, 0
2870                                                db 2, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0FFh
2871                                                db 0FFh, 0FFh, 0FFh, 0FFh, 4Eh,      4Fh, 20h, 4Eh, 41h, 4Dh
2872                                                db 45h,20h, 20h, 20h, 20h, 0, 0, 0, 0,      0, 46h,41h, 54h
2873                                                db 31h,32h, 20h, 20h, 20h, 0
2874
2875                                    bds3:               dw bds4
2876                                                dw 70h
2877                                                db 0
2878                                                db 0
2879                                    fdrive3:    dw 512
2880                                                db 0FFh, 1, 0, 2, 40h, 0, 68h, 1, 0, 2,     0, 9, 0, 1, 0
2881                                                db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 3, 20h, 0, 28h, 0
2882                                                db 0, 2, 1, 1, 0, 2, 0E0h, 0, 68h, 1, 0F0h, 2, 0, 9, 0
2883                                                db 2, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0FFh
2884                                                db 0FFh, 0FFh, 0FFh, 0FFh, 4Eh,      4Fh, 20h, 4Eh, 41h, 4Dh
2885                                                db 45h,20h, 20h, 20h, 20h, 0, 0, 0, 0,      0, 46h,41h, 54h
2886                                                db 31h,32h, 20h, 20h, 20h, 0
2887                                    ; ----
2888
2889                                    bds4:               dw 0FFFFh
2890                                                dw 70h
2891                                                db 0
2892                                                db 0
2893                                    fdrive4:    dw 512
2894                                                db 0FFh, 1, 0, 2, 40h, 0, 68h, 1, 0, 2,     0, 9, 0, 1, 0
2895                                                db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 3, 20h, 0, 28h, 0
2896                                                db 0, 2, 1, 1, 0, 2, 0E0h, 0, 68h, 1, 0F0h, 2, 0, 9, 0
2897                                                db 2, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0FFh
2898                                                db 0FFh, 0FFh, 0FFh, 0FFh, 4Eh,      4Fh, 20h, 4Eh, 41h, 4Dh
2899                                                db 45h,20h, 20h, 20h, 20h, 0, 0, 0, 0,      0, 46h,41h, 54h
2900                                                db 31h,32h, 20h, 20h, 20h, 0
2901
2902                                    ;--------------------------------------------------------------------------
2903
2904                                    sm92:               db 3                    ; .spf
2905                                                db 9                    ; .spt
2906                                                db 112  ; 70h           ; .cdire
2907                                                dw 1440 ; 2*9*80        ; .csec
2908                                                db 2                    ; .spau
2909                                                db 2                    ; .chead
2910
2911                                    %endif
2912
2913 0000047E 00                       keyrd_func: db 0
2914 0000047F 01                       keysts_func:        db 1
2915 00000480 00                       printdev:   db 0                    ; printer device index
2916
2917                                    wait_count: ;dw 4 dup(50h)          ; retry counts for printers
2918 00000481 5000<rep 4h>                          times 4 dw 50h          ; 19/10/2022
2919
2920 00000489 0000                     daycnt:             dw 0
2921 0000048B 00                       t_switch:   db 0                    ; flag for updating daycnt
2922 0000048C 00                       havecmosclock:      db 0
2923 0000048D 13                       base_century:       db 19
2924 0000048E 50                       base_year:  db 80
2925
2926 0000048F 1F                       month_tab:  db 31
2927 00000490 1C                       february:   db 28 ; 08/08/2023
2928 00000491 1F1E1F1E1F1F1E1F1E-                   db 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
2928 0000049A 1F
2929
2930                                    ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
2931                                    %if 0
2932                                    bintobcd:   dw bin_to_bcd           ; points to bin_to_bcd proc in msinit
2933                                                dw 70h ; 17/10/2022
2934                                    daycnttoday:        dw daycnt_to_day        ; points to daycnt_to_day in msinit
2935                                                dw 70h ; 17/10/2022
2936                                    %endif
2937
2938 0000049B 00                       set_id_flag:        db 0                    ; flag for getbp routine
2939
2940                                    ; 02/10/2023 - Retro DOS v5.0 IBMBIO.COM (Modified PCDOS 7.1)
2941                                    ;fat_12_id: db 'FAT12   ',0
2942                                    ;fat_16_id: db 'FAT16   ',0
2943                                    ;vol_no_name:       db 'NO NAME    ',0
2944                                    ;temp_h:    dw 0                    ; temporary for       32 bit calculation
2945
2946 0000049C 0000                     start_sec_h:        dw 0                    ; starting sector number high word
2947 0000049E 0000                     saved_word: dw 0                    ; tempory saving place for a word
2948 000004A0 0000                     multrk_flag:        dw 0
2949 000004A2 00                       ec35flag:   db 0                    ; flags for 3.5 inch disk drives
2950 000004A3 0000                     vretry_cnt: dw 0
2951 000004A5 0000                     soft_ecc_cnt:       dw 0
2952 000004A7 00                       multitrk_format_flag: db 0          ; multi track format request flag
2953 000004A8 0000                     xfer_seg:   dw 0                    ; temp for transfer segment
2954
2955                                    ; variables for msdioctl.asm module
2956
2957                                    ; tracktable contains a 4-tuples (c,h,r,n) for each sector in a track
2958                                    ; c = cylinder number,h = head number,r = sector id,n = bytes per sector
2959                                    ;   n       bytes per sector
2960                                    ;       ---  ----------------
```

```
2961                                    ;   0            128
2962                                    ;   1            256
2963                                    ;   2            512
2964                                    ;   3           1024
2965
2966                                    ;max_sectors_curr_sup equ 63            ; current maximum sec/trk that
2967                                    ;                                   ; we support (was 40 in dos 3.2)
2968
2969 000004AA 2400                      sectorspertrack: dw 36
2970 000004AC 00000102                  tracktable: db 0, 0, 1, 2
2971 000004B0 00000202                              db 0, 0, 2, 2
2972 000004B4 00000302                              db 0, 0, 3, 2
2973 000004B8 00000402                              db 0, 0, 4, 2
2974 000004BC 00000502                              db 0, 0, 5, 2
2975 000004C0 00000602                              db 0, 0, 6, 2
2976 000004C4 00000702                              db 0, 0, 7, 2
2977 000004C8 00000802                              db 0, 0, 8, 2
2978 000004CC 00000902                              db 0, 0, 9, 2
2979 000004D0 00000A02                              db 0, 0, 10, 2
2980 000004D4 00000B02                              db 0, 0, 11, 2
2981 000004D8 00000C02                              db 0, 0, 12, 2
2982 000004DC 00000D02                              db 0, 0, 13, 2
2983 000004E0 00000E02                              db 0, 0, 14, 2
2984 000004E4 00000F02                              db 0, 0, 15, 2
2985 000004E8 00001002                              db 0, 0, 16, 2
2986 000004EC 00001102                              db 0, 0, 17, 2
2987 000004F0 00001202                              db 0, 0, 18, 2
2988 000004F4 00001302                              db 0, 0, 19, 2
2989 000004F8 00001402                              db 0, 0, 20, 2
2990 000004FC 00001502                              db 0, 0, 21, 2
2991 00000500 00001602                              db 0, 0, 22, 2
2992 00000504 00001702                              db 0, 0, 23, 2
2993 00000508 00001802                              db 0, 0, 24, 2
2994 0000050C 00001902                              db 0, 0, 25, 2
2995 00000510 00001A02                              db 0, 0, 26, 2
2996 00000514 00001B02                              db 0, 0, 27, 2
2997 00000518 00001C02                              db 0, 0, 28, 2
2998 0000051C 00001D02                              db 0, 0, 29, 2
2999 00000520 00001E02                              db 0, 0, 30, 2
3000 00000524 00001F02                              db 0, 0, 31, 2
3001 00000528 00002002                              db 0, 0, 32, 2
3002 0000052C 00002102                              db 0, 0, 33, 2
3003 00000530 00002202                              db 0, 0, 34, 2
3004 00000534 00002302                              db 0, 0, 35, 2
3005 00000538 00002402                              db 0, 0, 36, 2
3006
3007                                    ; 02/10/2023 - Retro DOS v5.0 IBMBIO.COM (Modified PCDOS 7.1)
3008                                    ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:053Ch
3009
3010                                                ;times 108 db 0             ; 19/10/2022
3011                                                ;;db 108 dup(0)             ; 4*max_sectors_curr_sup - ($ -      tracktable) dup       (0)
3012                                            ; times ((4*63) - 144) db 0
3013                                    dskdrvs:
3014 0000053C [5803]                                dw fdrive1
3015 0000053E [EE03]                                dw fdrive2
3016
3017                                                ;dw 52 dup(0)
3018 00000540 00<rep 68h>                           times 104 db 0          ; times (((4*63)-144)-4) db 0
3019                                                                        ; 4*max_sectors_curr_sup-($-tracktable)-4 dup (0)
3020
3021                                    ;--------------------------------------------------------------------------------
3022
3023                                    ; this is a real ugly place to put this
3024                                    ; it should really go in the bds
3025
3026 000005A8 00                        mediatype:  db 0
3027 000005A9 00                        media_set_for_format: db 0          ; 1 if we have done an int 13h set media
3028                                                                        ; type for format call
3029 000005AA 00                        had_format_error: db 0                      ; 1 if the previous format operation
3030                                                                        ; failed.
3031
3032                                    ; temp disk base table. it holds the the current dpt which is then replaced by
3033                                    ; the one passed by "new roms" before we perform a format operation. the old
3034                                    ; dpt is restored in restoreolddpt. the first entry (disk_specify_1) is -1 if
3035                                    ; this table does not contain the previously saved dpt.
3036
3037 000005AB FFFFFFFF                  tempdpt:    dd 0FFFFFFFFh ; -1      ; temp disk base table
3038 000005AF FF                        model_byte: db 0FFh                 ; model byte set at init time
3039 000005B0 00                        secondary_model_byte: db 0
3040
3041 000005B1 00                        int19sem:   db 0                    ; indicate that all int 19h
3042                                                                        ; initialization is complete
3043
3044                                    ;; we assume the following remain contiguous and their order doesn't change
3045                                    ;i19_lst:
3046                                    ;   irp     aa,<02,08,09,0a,0b,0c,0d,0e,70,72,73,74,76,77>
3047                                    ;   public  int19old&aa
3048                                    ;           db      aa&h    ; store the number as a byte
3049                                    ;int19old&aa       dd      -1      ; original hardware int. vectors for int 19h.
3050                                    ;   endm
3051
3052                                    ; 21/10/2022
3053
3054 000005B2 02                        i19_lst:    db 2
3055                                                                        ; Int19old&aa
3056 000005B3 FFFFFFFF                  int19old02: dd 0FFFFFFFFh ; -1
3057 000005B7 08                                    db 8
3058 000005B8 FFFFFFFF                  int19old08: dd 0FFFFFFFFh           ; original hardware int. vectors for int 19h
3059 000005BC 09                                    db 9
3060 000005BD FFFFFFFF                  int19old09: dd 0FFFFFFFFh
3061 000005C1 0A                                    db 0Ah
3062 000005C2 FFFFFFFF                  int19old0A: dd 0FFFFFFFFh
3063 000005C6 0B                                    db 0Bh
3064 000005C7 FFFFFFFF                  int19old0B: dd 0FFFFFFFFh
3065 000005CB 0C                                    db 0Ch
3066 000005CC FFFFFFFF                  int19old0C: dd 0FFFFFFFFh
3067 000005D0 0D                                    db 0Dh
3068 000005D1 FFFFFFFF                  int19old0D: dd 0FFFFFFFFh
3069 000005D5 0E                                    db 0Eh
3070 000005D6 FFFFFFFF                  int19old0E: dd 0FFFFFFFFh
3071 000005DA 70                                    db 70h
3072 000005DB FFFFFFFF                  int19old70: dd 0FFFFFFFFh
3073 000005DF 72                                    db 72h
3074 000005E0 FFFFFFFF                  int19old72: dd 0FFFFFFFFh
3075 000005E4 73                                    db 73h
3076 000005E5 FFFFFFFF                  int19old73: dd 0FFFFFFFFh
3077 000005E9 74                                    db 74h
3078 000005EA FFFFFFFF                  int19old74: dd 0FFFFFFFFh
3079 000005EE 76                                    db 76h
3080 000005EF FFFFFFFF                  int19old76: dd 0FFFFFFFFh
3081 000005F3 77                                    db 77h
3082 000005F4 FFFFFFFF                  int19old77: dd 0FFFFFFFFh
3083
3084                                    ;num_i19    equ ($ - i19_lst)/5  ; 18/03/2019
```

```
                                ;--------------------------------------------------------------------------
                                ; 02/10/2023 - Retro DOS v5.0 IBMBIO.COM (Modified PCDOS 7.1)
                                ;
                                ;dskdrvs:   dw fdrive1
                                ;           dw fdrive2
                                ;           dw fdrive3
                                ;           dw fdrive4
                                ;
                                ;;M011 -- made all hard drive stuff variable
                                ;           ;dw 22 dup(0)         ; up to 26 drives for mini disks
                                ;           times 22 dw 0  ; 19/10/2022
                                ;--------------------------------------------------------------------------
                                ; 01/10/2022 - Retro DOS v4.0 (MSDOS v5.0 -actual-)
                                ; 30/12/2018 - Retro DOS v4.0 (MSDOS v6.21 -draft-)
                                ; 01/06/2018 - Retro DOS v3.0 (MSDOS v3.3)

                                ;variables for dynamic relocatable modules
                                ;these should be stay resident.

000005F8 00000000               int6c_ret_addr:    dd 0                 ; return address from int 6Ch
                                                                        ; for p12 machine
                                ; data structures for real-time date and time

000005FC 00000000               bin_date_time:     db 0, 0, 0, 0        ; century, year, month,      day

                                ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                %if 0
                                month_table:       dw 0                 ; january
                                           dw 31                 ; february
                                           dw 59
                                           dw 90
                                           dw 120
                                           dw 151
                                           dw 181
                                           dw 212
                                           dw 243
                                           dw 273
                                           dw 304
                                           dw 334                ; december
                                %endif

00000600 0000                   daycnt2:   dw 0
                                ; 08/08/2023
                                ;feb29:            db 0                 ; february 29 in a leap      year flag

                                ;--------------------------------------------------------------------------
                                ;
                                ; 01/10/2022 - (New/Actual) Retro DOS v4.0 (will run as MSDOS 5.0)
                                ; by Erdogan Tan (Istanbul) ! free source code !
                                ; 31/12/2018 - (old/draft) Retro DOS v4.0 (will/would run as MSDOS 6.21)

                                ; -------------------------------------------------------------------------

                                ;************************************************************************
                                ;*                                                                      *
                                ;*  Entry points into Bios_Code routines. The segment values   *
                                ;*    are plugged in by seg_reinit.                                     *
                                ;*                                                                      *
                                ;************************************************************************

                                ; 01/10/2022 - Retro DOS v4.0 - IO.SYS (MSDOS v5.0)
                                ; BIOSCODE_SEGMENT equ 2C7h
                                ; BIOSDATA_SEGMENT equ 70h ; KERNEL_SEGMENT equ 70h

                                ; 01/10/2022 - Erdogan Tan
                                ; (disassembled MSDOS 5.0 IO.SYS code here with fixed function/routine
                                ;  addresses, they will be changed to table labels later)

                                ; 09/12/2022
                                %if 0
                                cdev:              dw 43h,2C7h          ; chardev_entry
                                                                        ; at 2C7h:43h =       70h:25B3h
                                ttticks:   dw 396h, 2C7h                ; time_to_ticks
                                                                        ; at 2C7h:396h = 70h:2906h
                                bcode_i2f:  dw 1302h, 2C7h              ; i2f_handler
                                                                        ; at 2C7h:1302h       = 70h:3872h
                                i13x:              dw 154Bh, 2C7h       ; i13z
                                                                        ; at 2C7h:154Bh       = 70h:3ABBh
                                %endif

                                ; 30/12/2022
                                ; (IOSYSCODESEG is 2CCh for MSDOS 6.21 IO.SYS)

                                ; 02/10/2023 - Retro DOS v5.0 IBMBIO.COM (Modified PCDOS 7.1)
                                ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:0602h
                                ; (IOSYSCODESEG is 364h for PCDOS 7.1 IBMBIO.COM)

                                ; 09/12/2022
00000602 [4700]2D03             cdev:              dw chardev_entry, IOSYSCODESEG
00000606 [A003]2D03             ttticks:   dw time_to_ticks, IOSYSCODESEG
                                ; 07/08/2023
                                ;bcode_i2f: dw i2f_handler, IOSYSCODESEG
0000060A [5718]2D03             i13x:              dw i13z, IOSYSCODESEG

                                end_BC_entries:    ; 15/10/2022

                                ;************************************************************************
                                ;*                                                                      *
                                ;*  cbreak - break key handling - simply set altah=3 and iret  *
                                ;*                                                                      *
                                ;************************************************************************

                                cbreak:
0000060E 2EC606[0C00]03                    mov     byte [cs:altah], 3 ; break key handling
                                                                        ; indicate break key set
                                intret:
00000614 CF                                iret

                                ; =============== S U B   R O U T I N E =======================================


                                ;************************************************************************
                                ;*                                                                      *
                                ;*  strategy - store es:bx (device driver request packet)             *
                                ;*               away at [ptrsav] for next driver function call *
                                ;*                                                                      *
                                ;************************************************************************

                                strategy:   ; proc far
```

```
00000615 2E891E[1200]                   mov     [cs:ptrsav], bx ; store es:bx (device driver request packet)
                                                                ; away at [ptrsav] for next driver function call
0000061A 2E8C06[1400]                   mov     [cs:ptrsav+2], es
0000061F CB                             retf

                                ; ---------------------------------------------------------------------------

                                ;****************************************************************************
                                ;*                                                                          *
                                ;*  device driver entry points. these are the initial                       *
                                ;*    'interrupt' hooks out of the device driver chain.                     *
                                ;*    in the case of our resident drivers, they'll just                     *
                                ;*    stick a fake return address on the stack which                        *
                                ;*    points to dispatch tables and possibly some unit                      *
                                ;*    numbers, and then call through a common entry point                       *
                                ;*    which can take care of a20 switching                                  *
                                ;*                                                                          *
                                ;****************************************************************************

                                ; 01/10/2022 - Erdogan Tan
                                ; (disassembled MSDOS 5.0 IO.SYS code here with fixed table
                                ;  addresses, they will be changed to table labels later)

                                ; 09/12/2022

                                ; 02/10/2023 - Retro DOS v5.0
                                ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:0620h, BIOSCODE = 0364h)

                                con_entry:
00000620 E84000                         call    cdev_entry
                                ; ---------------------------------------------------------------------------
                                        ;dw 0E4h               ; con_table
00000623 [E400]                         dw con_table           ; 364h:0E4h (PCDOS 7.1)
                                                               ; 2C7h:0E4h = 70h:2654h
                                ; ---------------------------------------------------------------------------

                                prn0_entry:
00000625 E83B00                         call    cdev_entry
                                ; ---------------------------------------------------------------------------
                                        ;dw 0FBh               ; prn_table
00000628 [FB00]                         dw prn_table
                                                               ; 2C7h:0FBh = 70h:266Bh
0000062A 0000                           db 0, 0
                                ; ---------------------------------------------------------------------------

                                prn1_entry:
0000062C E83400                         call    cdev_entry
                                ; ---------------------------------------------------------------------------
                                        ;dw 0FBh               ; prn_table
0000062F [FB00]                         dw prn_table
                                                               ; 2C7h:0FBh = 70h:266Bh
00000631 0001                           db 0, 1
                                ; ---------------------------------------------------------------------------

                                prn2_entry:
00000633 E82D00                         call    cdev_entry
                                ; ---------------------------------------------------------------------------
                                        ;dw 0FBh               ; prn_table
00000636 [FB00]                         dw prn_table
                                                               ; 2C7h:0FBh = 70h:266Bh
00000638 0102                           db 1, 2
                                ; ---------------------------------------------------------------------------

                                prn3_entry:
0000063A E82600                         call    cdev_entry
                                ; ---------------------------------------------------------------------------
                                        ;dw 0FBh               ; prn_table
0000063D [FB00]                         dw prn_table
                                                               ; 2C7h:0FBh = 70h:266Bh
0000063F 0203                           db 2, 3
                                ; ---------------------------------------------------------------------------

                                aux0_entry:
00000641 E81F00                         call    cdev_entry
                                ; ---------------------------------------------------------------------------
                                        ;dw 130h               ; aux_table
00000644 [3001]                         dw aux_table
                                                               ; 2C7h:130h = 70h:26A0h
00000646 00                             db 0
                                ; ---------------------------------------------------------------------------

                                aux1_entry:
00000647 E81900                         call    cdev_entry
                                ; ---------------------------------------------------------------------------
                                        ;dw 130h               ; aux_table
0000064A [3001]                         dw aux_table           ; 364h:130h = 70h:3070h (PCDOS 7.1)
                                                               ; 2C7h:130h = 70h:26A0h
0000064C 01                             db 1
                                ; ---------------------------------------------------------------------------

                                aux2_entry:
0000064D E81300                         call    cdev_entry
                                ; ---------------------------------------------------------------------------
                                        ;dw 130h               ; aux_table
00000650 [3001]                         dw aux_table
                                                               ; 2C7h:130h = 70h:26A0h
00000652 02                             db 2
                                ; ---------------------------------------------------------------------------

                                aux3_entry:
00000653 E80D00                         call    cdev_entry
                                ; ---------------------------------------------------------------------------
                                        ;dw 130h               ; aux_table
00000656 [3001]                         dw aux_table
                                                               ; 2C7h:130h = 70h:26A0h
00000658 03                             db 3
                                ; ---------------------------------------------------------------------------

                                tim_entry:
00000659 E80700                         call    cdev_entry
                                ; ---------------------------------------------------------------------------
                                        ;dw 147h               ; tim_table
0000065C [4701]                         dw tim_table           ; 364h:147h = 70h:3087h (PCDOS 7.1)
                                                               ; 2C7h:147h = 70h:26B7h
                                ; ---------------------------------------------------------------------------

                                ; 15/10/2022
                                ;DSKTBL     equ dsktbl - DOSBIOSEG_2C7h   ; dsktbl - 2C70h
                                ; 09/12/2022
                                DSKTBL equ dsktbl

                                dsk_entry:
0000065E E80200                         call    cdev_entry
                                ; ---------------------------------------------------------------------------
```

```
                                        ;dw 4A2h                ; dsktbl
00000661 [6F05]                         dw DSKTBL               ; 09/12/2022
                                                                ; 2C7h:4A2h = 70h:2A12h
                                                                ; 02/10/2023 (PCDOS 7.1 IBMBIO.COM)
                                                                ; 364h:579h = 70h:34B9h

                                ; ============== S U B   R O U T I N E =======================================

                                ;****************************************************************************
                                ;*                                                                          *
                                ;*  Ensure A20 is enabled before jumping into code in HMA.                  *
                                ;*  This code assumes that if Segment of Device request packet is           *
                                ;*  DOS DATA segment then the Device request came from DOS & that           *
                                ;*  A20 is already on.                                                      *
                                ;*                                                                          *
                                ;****************************************************************************

                                cdev_entry: ; proc near
00000663 2E803E[0D00]00                 cmp     byte [cs:inHMA],0
00000669 740D                           jz      short ce_enter_codeseg
                                                        ; optimized for DOS in HMA
0000066B 50                             push    ax
0000066C 2EA1[0300]                     mov     ax,[cs:DosDataSg]
00000670 2E3906[1400]                   cmp     [cs:ptrsav+2],ax
00000675 58                             pop     ax
00000676 7505                           jnz     short not_from_dos
                                                        ; jump is coded this way to fall thru
                                                        ; in 99.99% of the cases
                                ce_enter_codeseg:
00000678 2EFF2E[0206]                   jmp     far [cs:cdev]
                                        ;jmp    dword ptr cs:cdev
                                ;---------------------------------------------------------------------------

                                not_from_dos:
0000067D E8AA00                         call    EnsureA20On
00000680 EBF6                           jmp     short ce_enter_codeseg

                                ;****************************************************************************
                                ;*                                                                          *
                                ;*  outchr - this is our int 29h handler. it writes the                     *
                                ;*     character in al on the display using int 10h ttywrite                *
                                ;*                                                                          *
                                ;****************************************************************************

                                    ; 17/07/2024
                                    ; 02/10/2023
                                    ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:0682h
                                outchr:
00000682 50                             push    ax              ; int 29h handler
00000683 56                             push    si
00000684 57                             push    di
00000685 55                             push    bp
00000686 53                             push    bx
                                        ;;;
                                        ; 02/10/2023 - Retro DOS v5.0 (Modified POCDOS 7.1)
                                        ;mov    ah,0Eh
                                        ;mov    bx,7
                                        ;int    10h     ; - VIDEO - WRITE CHARACTER AND      ADVANCE CURSOR (TTY WRITE)
                                        ;               ; AL = character, BH = display page (alpha modes)
                                        ;               ; BL = foreground color       (graphics modes)
                                        ; 17/07/2024
                                        ; 02/10/2023
00000687 1E                             push    ds ; *
00000688 31DB                           xor     bx,bx ; 0
0000068A 8EDB                           mov     ds,bx ; 0
0000068C B40E                           mov     ah,0Eh
0000068E B307                           mov     bl,7
00000690 2E381E[1208]                   cmp     [cs:IsWin386], bl ; (are we in) Windows ?
                                        ; 17/07/2024
                                        ;jnz    short win_outchr ; *
                                        ;push   ds ; *
                                        ;mov    ds,bx ; 0
                                        ;mov    ah,0Eh
                                        ;mov    bl,7
00000695 7508                           jnz     short win_outchr ; Running on Windows
00000697 9C                             pushf                   ; far call (simulate INT)
00000698 FA                             cli     ; disable interrupts
00000699 FF1E4000                       call    far [40h]       ; far call to INT 10h vector
                                        ; 17/07/2024
                                        ;pop    ds ; *
0000069D EB02                           jmp     short outchr_ok
                                win_outchr:
0000069F CD10                           int     10h
                                outchr_ok:
                                        ; 17/07/2024
000006A1 1F                             pop     ds ; *
                                        ;;;
000006A2 5B                             pop     bx
000006A3 5D                             pop     bp
000006A4 5F                             pop     di
000006A5 5E                             pop     si
000006A6 58                             pop     ax
000006A7 CF                             iret

                                ;---------------------------------------------------------------------------

                                    ; 02/10/2023 - Retro DOS v5.0
                                    ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:06A8h

000006A8 50                             db 50h ; P              ; 'PCI' signature
000006A9 43                             db 43h ; C
000006AA 49                             db 49h ; I

000006AB 00000000               Orig1A:         dd 0

                                ; ============== S U B R O U T I N E =======================================

                                    ; 02/10/2023 - Retro DOS v5.0
                                    ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:06AFh

                                Int1A:
000006AF 80FC04                         cmp     ah,4            ; (Y2K-fix)
000006B2 7405                           je      short int1a_1   ; Read the date from the computer's real-time clock
000006B4 2EFF2E[AB06]                   jmp     far [cs:Orig1A]
                                int1a_1:
000006B9 55                             push    bp
                                int1a_2:
000006BA 89E5                           mov     bp,sp
000006BC 55                             push    bp
000006BD 9C                             pushf
000006BE 2EFF1E[AB06]                   call    far [cs:Orig1A]
000006C3 7220                           jc      short int1a_4

                                        ;cmp    cl,0            ; Year (BCD)
```

```
                                        ; 02/10/2023
000006C5 08C9                           or      cl,cl
000006C7 7515                           jnz     short int1a_3
000006C9 80FD19                         cmp     ch,19h          ; Century (BCD)
000006CC 7510                           jne     short int1a_3
000006CE B520                           mov     ch,20h
000006D0 B405                           mov     ah,5            ; Set the date on the computer's real-time clock
000006D2 51                             push    cx
000006D3 52                             push    dx              ; dh = Month (BCD), dl = Day (BCD)
000006D4 9C                             pushf
000006D5 2EFF1E[AB06]                   call    far [cs:Orig1A]
000006DA 5A                             pop     dx
000006DB 59                             pop     cx
000006DC 7207                           jc      short int1a_4
                                int1a_3:
000006DE 5D                             pop     bp
000006DF 806606FE                       and     byte [bp+6],0FEh ; clear carry flag
000006E3 EB05                               jmp       short int1a_5
                                int1a_4:
000006E5 5D                             pop     bp
000006E6 804E0601                       or      byte [bp+6],1  ; set carry flag
                                int1a_5:
000006EA 5D                             pop     bp
000006EB CF                             iret

                                        ; 02/10/2023
000006EC 90                             nop     ; (not necessary, i have used this 'nop' to locate 'block13:'
                                                ; at BIOSDATA:06EDh, just as in the original PCDOS 7.1 IBMBIO.COM)

                                ;--------------------------------------------------------------------------

                                ;**************************************************************************
                                ;*                                                                        *
                                ;*  block13 - our int13 hooker                                            *
                                ;*                                                                        *
                                ;**************************************************************************

                                    ; 02/10/2023 - Retro DOS v5.0
                                    ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:06EDh

                                block13:
000006ED 2E803E[0D00]00                 cmp     byte [cs:inHMA],0
000006F3 7403                           jz      short skipa20

                                        ;call   IsA20Off        ; A20 Off?
                                        ;jnz    short skipa20
                                        ;call   EnableA20       ; assure a20 enabled
                                        ; 02/10/2023 -  Retro DOS v5.0 (Modified PCDOS 7.1)
000006F5 E83200                         call    EnsureA20On     ; assure a20 enabled
                                skipa20:
000006F8 2E8C1E[1C00]                   mov     [cs:i13_ds],ds ; save caller's ds for call-through
000006FD 9C                             pushf                   ; fake interrupt
000006FE 2EFF1E[0A06]                   call    far [cs:i13x]
                                        ;call   dword ptr cs:i13x
                                                                ; call through Bios_Code entry table
00000703 2E8E1E[1C00]                   mov     ds,[cs:i13_ds]
00000708 CA0200                         retf    2

                                ; =============== S U B   R O U T I N E =======================================

                                ; the int13 hook calls back here to call-through to the ROM
                                ; this is necessary because some people have extended their
                                ; ROM BIOSs to use ds as a parameter/result register and
                                ; our int13 hook relies heavily on ds to access Bios_Data

                                call_orig13:        ; proc far
0000070B 8E1E[1C00]                     mov     ds,[i13_ds]     ; get caller's ds register
0000070F 9C                             pushf                   ; simulate an int13
00000710 2EFF1E[B400]                   call    far [cs:Orig13]
                                        ;call   cs:Orig13
00000715 2E8C1E[1C00]                   mov     [cs:i13_ds],ds
0000071A 0E                             push    cs
0000071B 1F                             pop     ds              ; restore ds ->       Bios_Data before return

0000071C 9C                             pushf
                                        ; 10/12/2022
                                        ; ds = cs
0000071D 803E[0D00]00                   cmp     byte [inHMA],0 ; 16/10/2022
                                        ;cmp    byte [cs:inHMA],0
00000722 7403                           jz      short corig13_popf_retf
                                        ; 07/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                        ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:0725h
                                        ;call   IsA20Off
                                        ;jnz    short corig13_popf_retf
                                        ;call   EnableA20
00000724 E80300                         call    EnsureA20On ; 07/08/2023
                                corig13_popf_retf:
00000727 9D                             popf
                                        ; 20/09/2023
                                re_init:    ; 07/08/2023
00000728 CB                             retf

                                        ; 02/10/2023
00000729 90                             nop     ; (not necessary, i have used this 'nop' to locate 'EnsureA20On:'
                                                ; at BIOSDATA:072Ah, just as in the original PCDOS 7.1 IBMBIO.COM)

                                ;--------------------------------------------------------------------------

                                ; BIOSDATA:07BBh (MSDOS 6.21, IO.SYS)
                                ; BIOSDATA:07BBh (MSDOS 5.0, IO.SYS) ; 16/10/2022

                                ; 07/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                ;HiMem:             dd 0FFFF0090h
                                ;LoMem:             dd 80h

                                ; ---------------------------------------------------------------------------

                                ; =============== S U B   R O U T I N E =======================================


                                ;**************************************************************************
                                ;*                                                                        *
                                ;*  EnsureA20On - ensure that a20 is enabled if we're running             *
                                ;*    in the HMA before interrupt entry points into Bios_Code             *
                                ;*                                                                        *
                                ;**************************************************************************

                                EnsureA20On:        ; proc near
0000072A E80E00                         call    IsA20Off
                                        ;jz     short EnableA20
                                        ;retn
                                        ; 18/12/2022
0000072D 750B                           jnz     short A20On_retn

```

```
                                ; ============== S U B   R O U TI N E =======================================


                                EnableA20:  ; proc near
0000072F 50                                 push    ax
00000730 53                                 push    bx
00000731 B405                               mov     ah,5    ; local enable a20
                                            ;call   cs:xms
00000733 2EFF1E[0E00]                       call    far [cs:xms] ; 16/10/2022
00000738 5B                                 pop     bx
00000739 58                                 pop     ax
                                A20On_retn: ; 18/12/2022
0000073A C3                                 retn

                                ; ============== S U B   R O U TI N E =======================================


                                IsA20Off:   ; proc near
0000073B 1E                                 push    ds
0000073C 06                                 push    es
0000073D 51                                 push    cx
0000073E 56                                 push    si
0000073F 57                                 push    di
                                            ; 07/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                            ;lds    si,[cs:HiMem]
                                            ;les    di,[cs:LoMem]
                                            ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:0740h
00000740 31FF                               xor     di,di
00000742 8EC7                               mov     es,di
00000744 4F                                 dec     di
00000745 BE9000                             mov     si,90h  ; 0FFFFh:0090h ; HiMem
00000748 8EDF                               mov     ds,di
0000074A BF8000                             mov     di,80h  ; 0000h:0080h ; LoMem
                                            ; 02/10/2023 - Retro DOS v5.0 IBMBIO.COM (PCDOS 7.1)
                                            ; (following cpu instructions will be modified by 'SYSIN'
                                            ; if the cpu is a 386/32bit, for checking A20 line faster)
                                cpu386_cmpsd:
0000074D 90                                 nop
0000074E B90800                             mov     cx,8
00000751 F3A7                               repe cmpsw
                                                    ; zf = 0 -> A20 line is ON
                                                    ; zf = 1 -> A20 line is OFF
00000753 5F                                 pop     di
00000754 5E                                 pop     si
00000755 59                                 pop     cx
00000756 07                                 pop     es
00000757 1F                                 pop     ds
00000758 C3                                 retn

                                ; ---------------------------------------------------------------------------

                                ; 07/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                %if 0
                                DisableA20:
                                            push    ax
                                            push    bx
                                            mov     ah,6            ; local disable A20
                                            call    far [cs:xms]
                                            ;call   cs:xms
                                            pop     bx
                                            pop     ax
                                            retn
                                %endif

                                ; ---------------------------------------------------------------------------

                                ;****************************************************************************
                                ;*                                                                          *
                                ;*  int19 - bootstrap interrupt -- we must restore a bunch of the           *
                                ;*    interrupt vectors before resuming the original int19 code             *
                                ;*                                                                        *
                                ;****************************************************************************

                                            ; 02/10/2023 - Retro DOS v5.0
                                            ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:0759h
                                int19:
00000759 0E                                 push    cs
0000075A 1F                                 pop     ds
                                            ; 07/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                            ;mov    es,[zeroseg]    ; 16/10/2022
                                            ;mov    cx,5            ; NUMROMVECTORS
0000075B 31C9                               xor     cx,cx
0000075D 8EC1                               mov     es,cx
0000075F B105                               mov     cl,5
                                            ;mov    si,offset RomVectors
00000761 BE[0001]                           mov     si,RomVectors  ; 19/10/2022
                                next_int:
00000764 AC                                 lodsb                   ; get int number
00000765 98                                 cbw                     ; assume < 128
00000766 D1E0                               shl     ax,1
00000768 D1E0                               shl     ax,1            ; int * 4
                                            ; 07/08/2023
                                            ;mov    di,ax
                                            ;lodsw
                                            ;stosw
                                            ;lodsw
                                            ;stosw                  ; install the saved vector
                                            ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:076Ah
0000076A 97                                 xchg    ax,di
0000076B A5                                 movsw
0000076C A5                                 movsw
0000076D E2F5                               loop    next_int
                                            ;cmp    byte [int19sem], 0 ; 19/10/2022
0000076F 380E[B105]                         cmp     [int19sem], cl ; 0 ; 07/08/2023
00000773 7419                               jz      short doint19
00000775 BE[B205]                           mov     si,i19_lst      ; stacks code has changed these hardware interrupt vectors
                                                                    ; stkinit in sysinit1 will initialize int19oldxx values
                                            ;mov    cx,14           ; num_i19
                                            ; 07/08/2023
00000778 B10E                               mov     cl,14
                                i19_restore_loop:
0000077A AC                                 lodsb                   ; get interrupt       number
0000077B 98                                 cbw                     ; assume < 128
                                            ;mov    di,ax
                                            ;lodsw                  ; get original vector offset
                                            ;mov    bx,ax           ; save it
                                            ;lodsw
                                            ; 07/08/2023
0000077C 97                                 xchg    ax,di
0000077D AD                                 lodsw
0000077E 93                                 xchg    ax,bx
0000077F AD                                 lodsw
                                            ;cmp    bx,0FFFFh       ; check for 0ffffh (unlikely segment)
00000780 43                                 inc     bx ; 07/08/2023
```

```
3705 00000781 7409                              jz      short i19_restor_1 ; opt no need to check selector too
3706                                            ;cmp    ax,0FFFFh       ; opt 0ffffh is       unlikely offset
3707                                            ;jz     short i19_restor_1
3708 00000783 4B                                dec     bx ; 07/08/2023
3709 00000784 01FF                              add     di,di
3710 00000786 01FF                              add     di,di
3711 00000788 93                                xchg    ax,bx
3712 00000789 AB                                stosw
3713 0000078A 93                                xchg    ax,bx
3714 0000078B AB                                stosw                   ; put the vector back
3715                                    i19_restor_1:
3716 0000078C E2EC                              loop    i19_restore_loop
3717                                    doint19:
3718                                            ;cmp    byte [inHMA],0 ; ; Is dos running from      HMA
3719 0000078E 380E[0D00]                        cmp     [inHMA],cl ; 0 ; 07/08/2023
3720 00000792 7403                              jz      short SkipVDisk
3721 00000794 E82A00                            call    EraseVDiskHead ; Then erase our VDISK header at 1MB boundary
3722                                                                    ; Some m/c's (AST 386 & HP QS/16 do not clear
3723                                                                    ; the memory above 1MB during a      warm boot.
3724                                    SkipVDisk:
3725 00000797 CD19                              int     19h             ; DISK BOOT
3726                                                                    ; causes reboot       of disk system
3727
3728                                    ; =============== S U B   R O U T I N E =======================================
3729
3730                                    ;---------------------------------------------------------------------------
3731                                    ;
3732                                    ; procedure : int15
3733                                    ;
3734                                    ;           Int15 handler for recognizing ctrl-alt-del seq
3735                                    ;           If it recognizes ctrl-alt-del and if DOS was
3736                                    ;           is running high, it Erases the VDISK header
3737                                    ;           present at 1MB boundary
3738                                    ;
3739                                    ;---------------------------------------------------------------------------
3740
3741                                    ; 16/10/2022
3742                                    ;DELKEY             equ     53h
3743                                    ;ROMDATASEG equ     40h
3744                                    KBFLAG              equ     17h
3745                                    ;CTRLSTATE  equ     04h
3746                                    ;ALTSTATE   equ     08h
3747
3748                                                    ; 02/10/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
3749                                    Int15:
3750                                                    ;cmp    ax,4F00h+DELKEY
3751 00000799 3D534F                                   cmp     ax,4F53h        ; del keystroke ?
3752                                                    ; 02/10/2023 - Retro DOS v5.0
3753                                                    ; 07/08/2023
3754 0000079C 7405                                     jz      short int15_1
3755                                                    ;jnz    short Old15_j  ; 07/08/2023
3756                                    Old15_j:
3757 0000079E 2EFF2E[0B01]                             jmp     far [cs:Old15] ; 16/10/2022
3758
3759                                    ; ---------------------------------------------------------------------------
3760                                    int15_1:
3761 000007A3 1E                                       push    ds
3762 000007A4 50                                       push    ax
3763                                                    ; 07/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
3764                                                    ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:07A5h
3765                                                    ;mov    ax,40h          ; ROMDATASEG
3766                                                    ;mov    ds,ax
3767                                                    ;;mov   al,ds:17h       ; [KBFLAG]
3768                                                    ;; 16/10/2022
3769                                                    ;mov    al,[KBFLAG]
3770 000007A5 31C0                                     xor     ax,ax
3771 000007A7 8ED8                                     mov     ds,ax
3772 000007A9 A01704                                   mov     al,[0417h]      ; KBFLAG = 0417h (PCDOS 7.1 IBMBIO.COM)
3773 000007AC 240C                                     and     al,0Ch          ; (CTRLSTATE | ALTSTATE)
3774 000007AE 3C0C                                     cmp     al,0Ch          ; (CTRLSTATE | ALTSTATE)
3775 000007B0 750A                                     jnz     short int15_2
3776                                                    ; 07/08/2023
3777                                                    ;push   cs
3778                                                    ;pop    ds
3779                                                    ;cmp    byte [inHMA],0 ; is DOS running from HMA
3780 000007B2 2E3826[0D00]                             cmp     byte [cs:inHMA],ah ; 0
3781 000007B7 7403                                     jz      short int15_2
3782 000007B9 E80500                                   call    EraseVDiskHead
3783                                    int15_2:
3784 000007BC 58                                       pop     ax
3785 000007BD 1F                                       pop     ds
3786 000007BE F9                                       stc
3787                                                    ; 02/10/2023 - Retro DOS v5.0
3788 000007BF EBDD                                     jmp     short Old15_j
3789
3790                                                    ; 02/10/2023
3791                                    ;Old15_j:   ; 07/08/2023
3792                                    ;           jmp     far [cs:Old15] ; 16/10/2022
3793                                    ;           ;jmp    cs:Old15
3794
3795                                    ; =============== S U B   R O U T I N E =======================================
3796
3797                                    ;---------------------------------------------------------------------------
3798                                    ;
3799                                    ; procedure : EraseVDiskHead
3800                                    ;
3801                                    ;           Erases the VDisk Header present in the 1MB boundary
3802                                    ;
3803                                    ;---------------------------------------------------------------------------
3804
3805                                    EraseVDiskHead:     ; proc near
3806                                                    ; 07/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
3807                                                    ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:07C1h
3808                                                    ;push   ax
3809 000007C1 51                                       push    cx
3810 000007C2 57                                       push    di
3811 000007C3 06                                       push    es
3812 000007C4 E863FF                                   call    EnsureA20On
3813                                                    ;mov    ax,0FFFFh       ; HMA seg
3814                                                    ;mov    es,ax
3815                                                    ; 03/10/2023 - Retro DOS v5.0
3816 000007C7 6AFF                                     push    0FFFFh
3817 000007C9 07                                       pop     es
3818 000007CA BF1000                                   mov     di,10h          ; point to VDISK header
3819                                                    ; 07/08/2023
3820                                                    ;mov    cx,10h          ; size of vdisk       header
3821 000007CD 89F9                                     mov     cx,di ; 16
3822                                                    ; 03/10/2023
3823 000007CF 31C0                                     xor     ax,ax
3824                                                    ;inc    ax ; ax = 0
3825 000007D1 F3AB                                     rep stosw               ; clear it
3826 000007D3 07                                       pop     es
3827 000007D4 5F                                       pop     di
3828 000007D5 59                                       pop     cx
```

```
                                        ;pop    ax ; 07/08/2023
000007D6 C3                             retn

                                ; ----------------------------------------------------------------------------

                                ; 03/10/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
                                ; 17/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)

                                ; 09/12/2022
                                ;SYSINITSEG equ 46Dh  ; SYSINIT segment
                                ;DOSLOADSEG equ 83Fh  ; MSDOS.SYS (kernel) loading segment
                                ; (followings are in sysinit segment)
                                ;FTryToMovDOSHi    equ 0A84h ; (procedure in SYSINIT segment)
                                FTRYTOMOVDOSHI     equ FTryToMovDOSHi ; SYSINIT section
                                ;DEVICELIST equ 273h
                                DEVICELIST  equ DEVICE_LIST       ; SYSINIT section
                                ;MEMORYSIZE equ 292h
                                MEMORYSIZE  equ MEMORY_SIZE       ; SYSINIT section
                                ;DEFAULTDRIVE      equ 296h
                                DEFAULTDRIVE       equ DEFAULT_DRIVE ; SYSINIT section
                                ;;currentdoslocation equ 271h
                                ;CURRENTDOSLOCATION equ 271h
                                CURRENTDOSLOCATION equ CURRENT_DOS_LOCATION  ; SYSINIT section
                                ;SYSINITSTART      equ 267h
                                SYSINITSTART       equ SYSINIT  ; SYSINIT section
                                ; 18/10/2022
                                ;toomanydrivesflag equ 3FFh
                                TOOMANYDRIVESFLAG equ toomanydrivesflag ; SYSINIT section

                                ; ----------------------------------------------------------------------------

                                ; 03/10/2023 - Retro DOS v5.0 IBMBIO.COM (Modified PCDOS 7.1)
                                ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:07D7h

                                %if 1

000007D7 FFFF                   FreeHMAPtr: dw 0FFFFh
                                ;MoveDOSIntoHMA: dd 46D0A84h             ; FTryToMovDOSHi
                                                                ; (procedure in       SYSINIT segment)
                                ; 17/10/2022
000007D9 [E20B]                 MoveDOSIntoHMA:    dw FTRYTOMOVDOSHI     ; 09/12/2022
000007DB 0405                               dw SYSINITSEG         ; 08/08/2023
                                                                ; 0544h for PCDOS 7.1 IBMBIO.COM
                                                                ; 0473h for MSDOS 6.21 IO.SYS
                                ;SR;
                                ; A communication block has been setup between the DOS and the BIOS. All
                                ;the data starting from SysinitPresent will be part of the data block.
                                ;Right now, this is the only data being communicated. It can be expanded
                                ;later to add more stuff

000007DD 00                     SysinitPresent:    db 0

                                %endif

                                ; ----------------------------------------------------------------------------

                                ;****************************************************************************
                                ;*                                                                          *
                                ;*  the int2f handler chains up to Bios_Code through here.                  *
                                ;*    it returns through one of the three functions that follow.            *
                                ;*    notice that we'll assume we're being entered from DOS, so             *
                                ;*    that we're guaranteed to be A20 enabled if needed                     *
                                ;*                                                                          *
                                ;****************************************************************************

                                ; 07/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                %if 0       ; 20/09/2023
                                int_2f:
                                            jmp    far [cs:bcode_i2f] ; 16/10/2022
                                            ;jmp   dword ptr cs:bcode_i2f ; far [cs:bcode_i2f]

                                ; ----------------------------------------------------------------------------

                                ; re-enter here to transition out of hma mode and jmp to dsk_entry
                                ; note:  is it really necessary to transiton out and then back in?
                                ;    It's not as if this is a really speed critical function.
                                ;    might as well do whatever's most compact.

                                i2f_dskentry:
                                            jmp    dsk_entry

                                ; ----------------------------------------------------------------------------

                                ;****************************************************************************
                                ;*                                                                          *
                                ;*  re_init - called back by sysinit after a bunch of stuff                 *
                                ;*          is done. presently does nothing. affects no                     *
                                ;*          registers!                                                      *
                                ;*                                                                          *
                                ;****************************************************************************

                                ; 09/12/2022
                                ; re_init_:
                                re_init:                        ; called back by sysinit after
                                            retf                ; a bunch of stuff is done.
                                                                ; presently does nothing
                                %endif

                                ; ----------------------------------------------------------------------------

                                ;SR; WIN386 support

                                ; WIN386 instance data structure
                                ;
                                ; Here is a Win386 startup info structure which we set up and to which
                                ; we return a pointer when Win386 initializes.

000007DE 0300                   Win386_SI:  db 3,0              ; SI_Version
                                                                ; Startup Info for Win386
000007E0 00000000               SI_Next:    dd 0                ; pointer to next info structure
000007E4 00000000                           dd 0                ; a field we don't need
000007E8 00000000                           dd 0                ; another field       we don't need
000007EC [F007]                 SI_Instance:       dw Instance_Table
000007EE 7000                               dw 70h  ; Bios_Data    ; far pointer to instance table

                                ; This table gives Win386 the instance data in the BIOS and ROM-BIOS data
                                ; areas. Note that the address and size of the hardware stacks must
                                ; be calculated and inserted at boot time.

000007F0 00005000               Instance_Table:    dw 0,50h              ; print screen status...
000007F4 0200                               dw 2                ; ... 2 bytes
000007F6 0E005000                           dw 0Eh,50h          ; ROM Basic data...
000007FA 1400                               dw 14h              ; ... 14H bytes
000007FC [0C00]                             dw altah            ; a con device buffer...
```

```
000007FE 7000                           dw 70h                  ; Bios_Data segment
00000800 0100                           dw 1                    ; ... 1 byte

                                NextStack:

                                ; NOTE:  If stacks are disabled by STACKS=0,0, the following
                                ;   instance items WILL NOT be filled in by SYSINIT.
                                ;   That's just fine as long as these are the last items
                                ;   in the instance list since the first item is initialized
                                ;   to 0000 at load time.

00000802 00000000                       dw 0,0                  ; pointer to next stack      to be used...
00000806 0200                           dw 2                    ; ... 2 bytes
00000808 00000000               IT_StackLoc:    dd 0                    ; location of hardware stacks
0000080C 0000                   IT_StackSize:   dw 0                    ; size of hardware stacks
0000080E 00000000                       dd 0                    ; terminate the       instance table

                                                                ;SR;
00000812 00                     IsWin386:   db 0                ; Flag to indicate whether
                                                                ; Win386 is running or not
                                ;--------------------------------------------------------------------------

                                ; 09/12/2023 - Retro DOS v5.0 IBMBIO.COM (Modified PCDOS 7.1)
                                ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:0813h

                                ;This routine was originally in BIOS_CODE but this causes a lot of problems
                                ;when we call it including checking of A20. The code being only about
                                ;30 bytes, we might as well put it in BIOS_DATA

                                V86_Crit_SetFocus:
00000813 57                             push    di
00000814 06                             push    es
00000815 53                             push    bx
00000816 50                             push    ax
00000817 31FF                           xor     di,di
00000819 8EC7                           mov     es,di
0000081B BB1500                         mov     bx,15h          ; Device ID of DOSMGR device
0000081E B88416                         mov     ax,1684h        ; Get API entry       point
00000821 CD2F                           int     2Fh             ; - Multiplex -       MS WINDOWS - GET DEVICE       API ENTRY POINT
                                                                ; BX = virtual device (VxD) ID,       ES:DI =0000h:0000h
                                                                ; Return: ES:DI       -> VxD API entry point,       or 0:0 if the VxD does not
support an API
00000823 8CC0                           mov     ax, es
00000825 09F8                           or      ax, di
00000827 740A                           jz      short Skip      ; Here, es:di is address of API       routine.
                                                                ; Set up stack frame to       simulate a call.
00000829 0E                             push    cs
                                        ;;mov   ax,offset Skip
                                        ;mov    ax,Skip
                                        ;push   ax
                                        ; 03/10/2023 - Retro DOS v5.0
0000082A 68[3308]                       push    Skip
0000082D 06                             push    es
0000082E 57                             push    di              ; API far call address
0000082F B80100                         mov     ax,1            ; SetFocus function number
00000832 CB                             retf                    ; do the call
                                ;--------------------------------------------------------------------------

                                Skip:
00000833 58                             pop     ax
00000834 5B                             pop     bx
00000835 07                             pop     es
00000836 5F                             pop     di
00000837 CB                             retf

                                ;End WIN386 support

                                ; ------------------------------------------------------------------------

                                ; 03/10/2023 - Retro DOS v5.0 IBMBIO.COM (Modified PCDOS 7.1)
                                %if 0

                                FreeHMAPtr: dw 0FFFFh
                                ;MoveDOSIntoHMA: dd 46D0A84h             ; FTryToMovDOSHi
                                                                ; (procedure in       SYSINIT segment)
                                ; 17/10/2022
                                MoveDOSIntoHMA:    dw FTRYTOMOVDOSHI     ; 09/12/2022
                                        dw SYSINITSEG           ; 08/08/2023
                                                                ; 0544h for PCDOS 7.1 IBMBIO.COM
                                                                ; 0473h for MSDOS 6.21 IO.SYS
                                ;SR;
                                ; A communication block has been setup between the DOS and the BIOS. All
                                ;the data starting from SysinitPresent will be part of the data block.
                                ;Right now, this is the only data being communicated. It can be expanded
                                ;later to add more stuff

                                SysinitPresent:    db 0

                                endfloppy:  db 0, 0

                                %endif

                                    ; 03/10/2023 - Retro DOS v5.0 IBMBIO.COM

                                    endfloppy:
00000838 00                             db 0

                                    ; 03/10/2023

                                numxdiv     equ ($-BData_start)
                                numxmod     equ (numxdiv % 16)

                                %if (numxmod>0) & (numxmod<16)
00000839 00<rep 7h>                     times (16-numxmod) db 0
                                %endif

                                ; ------------------------------------------------------------------------

                                ; Bios_Data ends

                                ; Possibly disposable BIOS data
                                ; This data follows the   regular BIOS data,
                                ; and is part of the same group.

                                ; 03/10/2023 - Retro DOS v5.0 IBMBIO.COM
                                ;nul_vid:   db 'NO NAME    ',0    ; null volume id
                                ;tmp_vid:   db 'NO NAME    ',0    ; vid scratch buffer

                                ; 03/10/2023
00000840 4E4F204E414D452020-    tmp_vid:    db 'NO NAME    '
00000849 2020

                                harddrv:    db 80h
```

```
                                end96tpi:

                                ; 03/10/2023 - Retro DOS v5.0 IBMBIO.COM (Modified PCDOS 7.1)
                                ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:084Ch ('bdss:' address)

                                ;;**************************************************************************
                                ;;memory allocation for bdss
                                ;;**************************************************************************
                                ;
                                ;;max_mini_dsk_num equ 23 ; max # of mini disk ibmbio can support
                                ;
                                ;;bdss     BDS_STRUC (2+max_mini_dsk_num) dup (<>)     ; currently max. 25
                                ;
                                ;bdss:     times BDS.size*(2+max_mini_dsk_num) db 0


                                ; 09/12/2023
                                %if 1
                                ; Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM BDS structure (FAT32 adaptation)

0000084C FFFF                   bdss:             dw 0FFFFh             ; ...
                                                  ; max_mini_dsk_num equ 23
                                                  ; BDS_STRUC (2+max_mini_dsk_num) dup (<>)
                                                  ; currently max. 25
                                                  ; (MSDOS 6 BDS structure size = 100 bytes)
                                                  ; (PCDOS 7.1 BDS structure size = 150 bytes)
                                                  ; BDS.link
0000084E 0000                           dw 0
00000850 50                             db 80                 ; BDS.drivenum
00000851 03                             db 3                  ; BDS.drivelet
00000852 0002                           dw 512                ; BDS.BPB (BDS offset 6)
                                                              ; 53 bytes BPB for FAT32 fs
                                                              ; 25 bytes BPB for FAT16 and FAT12 fs
                                                              ; .bytespersec
00000854 01                             db 1                  ; .secperclus
00000855 0100                           dw 1                  ; .resectors
00000857 02                             db 2                  ; .fats
00000858 1000                           dw 16                 ; .direntries
0000085A 0000                           dw 0                  ; .totalsec16
0000085C F8                             db 0F8h               ; .media
0000085D 0100                           dw 1                  ; .fatsecs16
0000085F 0000                           dw 0                  ; .secpertrack
00000861 0000                           dw 0                  ; .heads
00000863 00000000                       dd 0                  ; .hiddensectors
00000867 00000000                       dd 0                  ; .totalsecs32
                                                              ; (End of FAT12/FAT16 BPB)
                                                              ;
                                                              ; FAT32 extensions to BDS
0000086B 00000000                       dd 0                  ; .fatsecs32 ; BPB_FATSz32 (BDS offset 31)
0000086F 0000                           dw 0                  ; .extflags ; BPB_ExtFlags
00000871 0000                           dw 0                  ; .fsver ; BPB_FSVer
00000873 00000000                       dd 0                  ; .rootdirclust ; BPB_RootClus (BDS offset 39)
00000877 FFFF                           dw 0FFFFh             ; .fsinfo ; BPB_FSInfo ; initialized to -1
00000879 FFFF                           dw 0FFFFh             ; .bkbootsec ; BPB_BkBootSec ; initialized to -1
0000087B 00<rep Ch>                     times 12 db 0         ; .reserved ; BPB_Reserved (12 zero bytes)
00000887 00                             db 0                  ; BDS.fatsiz (BDS offset 59)
00000888 0000                           dw 0                  ; BDS.opcnt
0000088A 03                             db 3
0000088B 2000                           dw 20h                ; BDS.flags (BDS offset 63)
0000088D 2800                           dw 40
0000088F 00<rep 25h>                    times 37 db 0
000008B4 FFFFFFFF                       dd 0FFFFFFFFh
000008B8 00<rep Ch>                     times 12 db 0
000008C4 FF                             db -1                 ; BDS.track (BDS offset 120)
000008C5 0100                           dw 1                  ; BDS.tim_lo ; BDS.bdsm_ismini
000008C7 0000                           dw 0                  ; BDS.tim_hi
000008C9 4E4F204E414D452020-            db 'NO NAME    ',0    ; BDS.volid
000008D2 202000
000008D5 00000000                       dd 0                  ; BDS.vol_serial (BDS offset 137)
000008D9 464154313220202000             db 'FAT12   ',0             ; BDS.filesys_id
000008E2 FFFF                   bds_1:          dw 0FFFFh
000008E4 000050030002010100-            db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
000008ED 0210000000F8
000008F3 010000000000000000-            db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
000008FC 000000000000000000
00000905 00000000000000000FF-           db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
0000090E FFFFFF0000
00000913 000000000000000000-            db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
0000091C 0000000003200028
00000924 000000000000000000-            db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0000092D 000000000000000000
00000936 000000000000000000-            db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0000093F 000000000000000000
00000948 0000FFFFFFFF000000-            db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
00000951 0000000000
00000956 00000000FF01000000-            db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
0000095F 4E4F204E41
00000964 4D4520202020000000-            db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
0000096D 00004641
00000971 54313220202000                 db 54h, 31h, 32h, 20h, 20h, 20h, 0
00000978 FFFF                   bds_2:          dw 0FFFFh
0000097A 000050030002010100-            db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
00000983 0210000000F8
00000989 010000000000000000-            db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
00000992 000000000000000000
0000099B 00000000000000000FF-           db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
000009A4 FFFFFF0000
000009A9 000000000000000000-            db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
000009B2 0000000003200028
000009BA 000000000000000000-            db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
000009C3 000000000000000000
000009CC 000000000000000000-            db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
000009D5 000000000000000000
000009DE 0000FFFFFFFF000000-            db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
000009E7 0000000000
000009EC 00000000FF01000000-            db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
000009F5 4E4F204E41
000009FA 4D4520202020000000-            db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
00000A03 00004641
00000A07 54313220202000                 db 54h, 31h, 32h, 20h, 20h, 20h, 0
00000A0E FFFF                   bds_3:          dw 0FFFFh
00000A10 000050030002010100-            db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
00000A19 0210000000F8
00000A1F 010000000000000000-            db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
00000A28 000000000000000000
00000A31 00000000000000000FF-           db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
00000A3A FFFFFF0000
00000A3F 000000000000000000-            db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
00000A48 0000000003200028
00000A50 000000000000000000-            db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
00000A59 000000000000000000
00000A62 000000000000000000-            db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
00000A6B 000000000000000000
00000A74 0000FFFFFFFF000000-            db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
```

```
4173 00000A7D 0000000000
4174 00000A82 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4174 00000A8B 4E4F204E41
4175 00000A90 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4175 00000A99 00004641
4176 00000A9D 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4177 00000AA4 FFFF                  bds_4:          dw 0FFFFh
4178 00000AA6 000050030002010100-               db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4178 00000AAF 0210000000F8
4179 00000AB5 0100000000000000000-              db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4179 00000ABE 0000000000000000
4180 00000AC7 00000000000000000FF-              db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4180 00000AD0 FFFFFF0000
4181 00000AD5 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4181 00000ADE 0000000003200028
4182 00000AE6 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4182 00000AEF 0000000000000000
4183 00000AF8 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4183 00000B01 0000000000000000
4184 00000B0A 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4184 00000B13 0000000000
4185 00000B18 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4185 00000B21 4E4F204E41
4186 00000B26 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4186 00000B2F 00004641
4187 00000B33 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4188 00000B3A FFFF                              dw 0FFFFh
4189 00000B3C 000050030002010100-               db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4189 00000B45 0210000000F8
4190 00000B4B 0100000000000000000-              db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4190 00000B54 0000000000000000
4191 00000B5D 00000000000000000FF-              db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4191 00000B66 FFFFFF0000
4192 00000B6B 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4192 00000B74 0000000003200028
4193 00000B7C 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4193 00000B85 0000000000000000
4194 00000B8E 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4194 00000B97 0000000000000000
4195 00000BA0 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4195 00000BA9 0000000000
4196 00000BAE 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4196 00000BB7 4E4F204E41
4197 00000BBC 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4197 00000BC5 00004641
4198 00000BC9 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4199 00000BD0 FFFF                              dw 0FFFFh
4200 00000BD2 000050030002010100-               db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4200 00000BDB 0210000000F8
4201 00000BE1 0100000000000000000-              db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4201 00000BEA 0000000000000000
4202 00000BF3 00000000000000000FF-              db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4202 00000BFC FFFFFF0000
4203 00000C01 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4203 00000C0A 0000000003200028
4204 00000C12 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4204 00000C1B 0000000000000000
4205 00000C24 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4205 00000C2D 0000000000000000
4206 00000C36 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4206 00000C3F 0000000000
4207 00000C44 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4207 00000C4D 4E4F204E41
4208 00000C52 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4208 00000C5B 00004641
4209 00000C5F 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4210 00000C66 FFFF                              dw 0FFFFh
4211 00000C68 000050030002010100-               db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4211 00000C71 0210000000F8
4212 00000C77 0100000000000000000-              db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4212 00000C80 0000000000000000
4213 00000C89 00000000000000000FF-              db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4213 00000C92 FFFFFF0000
4214 00000C97 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4214 00000CA0 0000000003200028
4215 00000CA8 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4215 00000CB1 0000000000000000
4216 00000CBA 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4216 00000CC3 0000000000000000
4217 00000CCC 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4217 00000CD5 0000000000
4218 00000CDA 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4218 00000CE3 4E4F204E41
4219 00000CE8 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4219 00000CF1 00004641
4220 00000CF5 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4221 00000CFC FFFF                              dw 0FFFFh
4222 00000CFE 000050030002010100-               db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4222 00000D07 0210000000F8
4223 00000D0D 0100000000000000000-              db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4223 00000D16 0000000000000000
4224 00000D1F 00000000000000000FF-              db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4224 00000D28 FFFFFF0000
4225 00000D2D 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4225 00000D36 0000000003200028
4226 00000D3E 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4226 00000D47 0000000000000000
4227 00000D50 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4227 00000D59 0000000000000000
4228 00000D62 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4228 00000D6B 0000000000
4229 00000D70 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4229 00000D79 4E4F204E41
4230 00000D7E 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4230 00000D87 00004641
4231 00000D8B 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4232 00000D92 FFFF                              dw 0FFFFh
4233 00000D94 000050030002010100-               db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4233 00000D9D 0210000000F8
4234 00000DA3 0100000000000000000-              db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4234 00000DAC 0000000000000000
4235 00000DB5 00000000000000000FF-              db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4235 00000DBE FFFFFF0000
4236 00000DC3 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4236 00000DCC 0000000003200028
4237 00000DD4 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4237 00000DDD 0000000000000000
4238 00000DE6 0000000000000000000-              db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4238 00000DEF 0000000000000000
4239 00000DF8 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4239 00000E01 0000000000
4240 00000E06 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4240 00000E0F 4E4F204E41
4241 00000E14 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
```

```
4241 00000E1D 00004641
4242 00000E21 54313220202000                  db 54h, 31h, 32h, 20h, 20h, 20h, 0
4243 00000E28 FFFF                            dw 0FFFFh
4244 00000E2A 000050030002010100-             db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4244 00000E33 0210000000F8
4245 00000E39 010000000000000000-             db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4245 00000E42 0000000000000000
4246 00000E4B 00000000000000FF-               db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4246 00000E54 FFFFFF0000
4247 00000E59 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4247 00000E62 0000000003200028
4248 00000E6A 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4248 00000E73 0000000000000000
4249 00000E7C 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4249 00000E85 0000000000000000
4250 00000E8E 0000FFFFFFFF000000-             db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4250 00000E97 0000000000
4251 00000E9C 00000000FF01000000-             db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4251 00000EA5 4E4F204E41
4252 00000EAA 4D4520202020000000-             db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4252 00000EB3 00004641
4253 00000EB7 54313220202000                  db 54h, 31h, 32h, 20h, 20h, 20h, 0
4254 00000EBE FFFF                            dw 0FFFFh
4255 00000EC0 000050030002010100-             db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4255 00000EC9 0210000000F8
4256 00000ECF 010000000000000000-             db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4256 00000ED8 0000000000000000
4257 00000EE1 00000000000000FF-               db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4257 00000EEA FFFFFF0000
4258 00000EEF 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4258 00000EF8 0000000003200028
4259 00000F00 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4259 00000F09 0000000000000000
4260 00000F12 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4260 00000F1B 0000000000000000
4261 00000F24 0000FFFFFFFF000000-             db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4261 00000F2D 0000000000
4262 00000F32 00000000FF01000000-             db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4262 00000F3B 4E4F204E41
4263 00000F40 4D4520202020000000-             db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4263 00000F49 00004641
4264 00000F4D 54313220202000                  db 54h, 31h, 32h, 20h, 20h, 20h, 0
4265 00000F54 FFFF                            dw 0FFFFh
4266 00000F56 000050030002010100-             db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4266 00000F5F 0210000000F8
4267 00000F65 010000000000000000-             db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4267 00000F6E 0000000000000000
4268 00000F77 00000000000000FF-               db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4268 00000F80 FFFFFF0000
4269 00000F85 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4269 00000F8E 0000000003200028
4270 00000F96 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4270 00000F9F 0000000000000000
4271 00000FA8 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4271 00000FB1 0000000000000000
4272 00000FBA 0000FFFFFFFF000000-             db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4272 00000FC3 0000000000
4273 00000FC8 00000000FF01000000-             db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4273 00000FD1 4E4F204E41
4274 00000FD6 4D4520202020000000-             db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4274 00000FDF 00004641
4275 00000FE3 54313220202000                  db 54h, 31h, 32h, 20h, 20h, 20h, 0
4276 00000FEA FFFF                            dw 0FFFFh
4277 00000FEC 000050030002010100-             db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4277 00000FF5 0210000000F8
4278 00000FFB 010000000000000000-             db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4278 00001004 0000000000000000
4279 0000100D 00000000000000FF-               db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4279 00001016 FFFFFF0000
4280 0000101B 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4280 00001024 0000000003200028
4281 0000102C 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4281 00001035 0000000000000000
4282 0000103E 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4282 00001047 0000000000000000
4283 00001050 0000FFFFFFFF000000-             db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4283 00001059 0000000000
4284 0000105E 00000000FF01000000-             db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4284 00001067 4E4F204E41
4285 0000106C 4D4520202020000000-             db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4285 00001075 00004641
4286 00001079 54313220202000                  db 54h, 31h, 32h, 20h, 20h, 20h, 0
4287 00001080 FFFF                            dw 0FFFFh
4288 00001082 000050030002010100-             db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4288 0000108B 0210000000F8
4289 00001091 010000000000000000-             db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4289 0000109A 0000000000000000
4290 000010A3 00000000000000FF-               db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4290 000010AC FFFFFF0000
4291 000010B1 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4291 000010BA 0000000003200028
4292 000010C2 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4292 000010CB 0000000000000000
4293 000010D4 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4293 000010DD 0000000000000000
4294 000010E6 0000FFFFFFFF000000-             db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4294 000010EF 0000000000
4295 000010F4 00000000FF01000000-             db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4295 000010FD 4E4F204E41
4296 00001102 4D4520202020000000-             db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4296 0000110B 00004641
4297 0000110F 54313220202000                  db 54h, 31h, 32h, 20h, 20h, 20h, 0
4298 00001116 FFFF                            dw 0FFFFh
4299 00001118 000050030002010100-             db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4299 00001121 0210000000F8
4300 00001127 010000000000000000-             db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4300 00001130 0000000000000000
4301 00001139 00000000000000FF-               db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4301 00001142 FFFFFF0000
4302 00001147 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4302 00001150 0000000003200028
4303 00001158 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4303 00001161 0000000000000000
4304 0000116A 000000000000000000-             db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4304 00001173 0000000000000000
4305 0000117C 0000FFFFFFFF000000-             db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4305 00001185 0000000000
4306 0000118A 00000000FF01000000-             db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4306 00001193 4E4F204E41
4307 00001198 4D4520202020000000-             db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4307 000011A1 00004641
4308 000011A5 54313220202000                  db 54h, 31h, 32h, 20h, 20h, 20h, 0
4309 000011AC FFFF                            dw 0FFFFh
4310 000011AE 000050030002010100-             db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
```

```
4310 000011B7 0210000000F8
4311 000011BD 010000000000000000-               db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4311 000011C6 0000000000000000
4312 000011CF 00000000000000FF-                 db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4312 000011D8 FFFFFF0000
4313 000011DD 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4313 000011E6 0000000003200028
4314 000011EE 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4314 000011F7 0000000000000000
4315 00001200 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4315 00001209 0000000000000000
4316 00001212 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4316 0000121B 0000000000
4317 00001220 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4317 00001229 4E4F204E41
4318 0000122E 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4318 00001237 00004641
4319 0000123B 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4320 00001242 FFFF                              dw 0FFFFh
4321 00001244 000050030002010100-               db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4321 0000124D 0210000000F8
4322 00001253 010000000000000000-               db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4322 0000125C 0000000000000000
4323 00001265 00000000000000FF-                 db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4323 0000126E FFFFFF0000
4324 00001273 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4324 0000127C 0000000003200028
4325 00001284 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4325 0000128D 0000000000000000
4326 00001296 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4326 0000129F 0000000000000000
4327 000012A8 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4327 000012B1 0000000000
4328 000012B6 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4328 000012BF 4E4F204E41
4329 000012C4 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4329 000012CD 00004641
4330 000012D1 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4331 000012D8 FFFF                              dw 0FFFFh
4332 000012DA 000050030002010100-               db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4332 000012E3 0210000000F8
4333 000012E9 010000000000000000-               db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4333 000012F2 0000000000000000
4334 000012FB 00000000000000FF-                 db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4334 00001304 FFFFFF0000
4335 00001309 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4335 00001312 0000000003200028
4336 0000131A 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4336 00001323 0000000000000000
4337 0000132C 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4337 00001335 0000000000000000
4338 0000133E 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4338 00001347 0000000000
4339 0000134C 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4339 00001355 4E4F204E41
4340 0000135A 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4340 00001363 00004641
4341 00001367 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4342 0000136E FFFF                              dw 0FFFFh
4343 00001370 000050030002010100-               db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4343 00001379 0210000000F8
4344 0000137F 010000000000000000-               db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4344 00001388 0000000000000000
4345 00001391 00000000000000FF-                 db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4345 0000139A FFFFFF0000
4346 0000139F 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4346 000013A8 0000000003200028
4347 000013B0 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4347 000013B9 0000000000000000
4348 000013C2 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4348 000013CB 0000000000000000
4349 000013D4 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4349 000013DD 0000000000
4350 000013E2 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4350 000013EB 4E4F204E41
4351 000013F0 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4351 000013F9 00004641
4352 000013FD 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4353 00001404 FFFF                              dw 0FFFFh
4354 00001406 000050030002010100-               db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4354 0000140F 0210000000F8
4355 00001415 010000000000000000-               db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4355 0000141E 0000000000000000
4356 00001427 00000000000000FF-                 db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4356 00001430 FFFFFF0000
4357 00001435 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4357 0000143E 0000000003200028
4358 00001446 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4358 0000144F 0000000000000000
4359 00001458 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4359 00001461 0000000000000000
4360 0000146A 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4360 00001473 0000000000
4361 00001478 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4361 00001481 4E4F204E41
4362 00001486 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4362 0000148F 00004641
4363 00001493 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4364 0000149A FFFF                              dw 0FFFFh
4365 0000149C 000050030002010100-               db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4365 000014A5 0210000000F8
4366 000014AB 010000000000000000-               db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4366 000014B4 0000000000000000
4367 000014BD 00000000000000FF-                 db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4367 000014C6 FFFFFF0000
4368 000014CB 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4368 000014D4 0000000003200028
4369 000014DC 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4369 000014E5 0000000000000000
4370 000014EE 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4370 000014F7 0000000000000000
4371 00001500 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4371 00001509 0000000000
4372 0000150E 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4372 00001517 4E4F204E41
4373 0000151C 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4373 00001525 00004641
4374 00001529 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4375 00001530 FFFF                              dw 0FFFFh
4376 00001532 000050030002010100-               db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4376 0000153B 0210000000F8
4377 00001541 010000000000000000-               db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4377 0000154A 0000000000000000
4378 00001553 00000000000000FF-                 db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
```

```
4378 0000155C FFFFFF0000
4379 00001561 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4379 0000156A 0000000003200028
4380 00001572 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4380 0000157B 0000000000000000
4381 00001584 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4381 0000158D 0000000000000000
4382 00001596 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4382 0000159F 0000000000
4383 000015A4 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4383 000015AD 4E4F204E41
4384 000015B2 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4384 000015BB 00004641
4385 000015BF 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4386 000015C6 FFFF                              dw 0FFFFh
4387 000015C8 000050030002010100-               db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4387 000015D1 0210000000F8
4388 000015D7 010000000000000000-               db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4388 000015E0 0000000000000000
4389 000015E9 0000000000000000FF-               db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4389 000015F2 FFFFFF0000
4390 000015F7 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4390 00001600 0000000003200028
4391 00001608 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4391 00001611 0000000000000000
4392 0000161A 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4392 00001623 0000000000000000
4393 0000162C 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4393 00001635 0000000000
4394 0000163A 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4394 00001643 4E4F204E41
4395 00001648 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4395 00001651 00004641
4396 00001655 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4397 0000165C FFFF                  bds_24:             dw 0FFFFh
4398 0000165E 000050030002010100-               db 0, 0, 50h, 3, 0, 2, 1, 1, 0, 2, 10h, 0, 0, 0, 0F8h
4398 00001667 0210000000F8
4399 0000166D 010000000000000000-               db 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4399 00001676 0000000000000000
4400 0000167F 0000000000000000FF-               db 0, 0, 0, 0, 0, 0, 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0
4400 00001688 FFFFFF0000
4401 0000168D 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 20h, 0, 28h
4401 00001696 0000000003200028
4402 0000169E 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4402 000016A7 0000000000000000
4403 000016B0 000000000000000000-               db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
4403 000016B9 0000000000000000
4404 000016C2 0000FFFFFFFF000000-               db 0, 0, 0FFh, 0FFh, 0FFh, 0FFh, 0, 0, 0, 0, 0, 0, 0, 0
4404 000016CB 0000000000
4405 000016D0 00000000FF01000000-               db 0, 0, 0, 0, 0FFh, 1, 0, 0, 0, 4Eh, 4Fh, 20h, 4Eh, 41h
4405 000016D9 4E4F204E41
4406 000016DE 4D4520202020000000-               db 4Dh, 45h, 20h, 20h, 20h, 20h, 0, 0, 0, 0, 0, 46h, 41h
4406 000016E7 00004641
4407 000016EB 54313220202000                    db 54h, 31h, 32h, 20h, 20h, 20h, 0
4408
4409                                    %endif
4410
4411                                    ; 09/12/2023
4412                                    %if 0
4413                                    ; Retro DOS v4.2 (MSDOS 6.22) IO.SYS BDS structure
4414
4415                                    bdss:               dw 0FFFFh
4416                                                db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h, 0, 0, 0, 0F8h
4417                                                db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4418                                                db 20h, 0, 28h, 0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0
4419                                                db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4420                                                db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4421                                                db 20h, 20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4422                                                db 32h, 20h, 20h, 20h, 0
4423                                                dw 0FFFFh
4424                                                db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h, 0, 0, 0, 0F8h
4425                                                db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4426                                                db 20h, 0, 28h, 0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0
4427                                                db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4428                                                db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4429                                                db 20h, 20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4430                                                db 32h, 20h, 20h, 20h, 0
4431                                                dw 0FFFFh
4432                                                db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h, 0, 0, 0, 0F8h
4433                                                db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4434                                                db 20h, 0, 28h, 0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0
4435                                                db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4436                                                db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4437                                                db 20h, 20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4438                                                db 32h, 20h, 20h, 20h, 0
4439                                                dw 0FFFFh
4440                                                db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h, 0, 0, 0, 0F8h
4441                                                db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4442                                                db 20h, 0, 28h, 0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0
4443                                                db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4444                                                db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4445                                                db 20h, 20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4446                                                db 32h, 20h, 20h, 20h, 0
4447                                                dw 0FFFFh
4448                                                db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h, 0, 0, 0, 0F8h
4449                                                db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4450                                                db 20h, 0, 28h, 0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0
4451                                                db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4452                                                db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4453                                                db 20h, 20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4454                                                db 32h, 20h, 20h, 20h, 0
4455                                                dw 0FFFFh
4456                                                db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h, 0, 0, 0, 0F8h
4457                                                db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4458                                                db 20h, 0, 28h, 0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0
4459                                                db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4460                                                db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4461                                                db 20h, 20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4462                                                db 32h, 20h, 20h, 20h, 0
4463                                                dw 0FFFFh
4464                                                db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h, 0, 0, 0, 0F8h
4465                                                db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4466                                                db 20h, 0, 28h, 0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0
4467                                                db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4468                                                db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4469                                                db 20h, 20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4470                                                db 32h, 20h, 20h, 20h, 0
4471                                                dw 0FFFFh
4472                                                db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h, 0, 0, 0, 0F8h
4473                                                db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4474                                                db 20h, 0, 28h, 0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0
4475                                                db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4476                                                db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4477                                                db 20h, 20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
```

```
4478                    db 32h,20h, 20h, 20h, 0
4479                    dw 0FFFFh
4480                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4481                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4482                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4483                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4484                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4485                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4486                    db 32h,20h, 20h, 20h, 0
4487                    dw 0FFFFh
4488                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4489                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4490                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4491                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4492                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4493                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4494                    db 32h,20h, 20h, 20h, 0
4495                    dw 0FFFFh
4496                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4497                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4498                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4499                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4500                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4501                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4502                    db 32h,20h, 20h, 20h, 0
4503                    dw 0FFFFh
4504                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4505                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4506                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4507                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4508                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4509                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4510                    db 32h,20h, 20h, 20h, 0
4511                    dw 0FFFFh
4512                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4513                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4514                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4515                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4516                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4517                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4518                    db 32h,20h, 20h, 20h, 0
4519                    dw 0FFFFh
4520                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4521                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4522                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4523                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4524                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4525                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4526                    db 32h,20h, 20h, 20h, 0
4527                    dw 0FFFFh
4528                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4529                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4530                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4531                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4532                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4533                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4534                    db 32h,20h, 20h, 20h, 0
4535                    dw 0FFFFh
4536                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4537                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4538                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4539                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4540                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4541                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4542                    db 32h,20h, 20h, 20h, 0
4543                    dw 0FFFFh
4544                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4545                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4546                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4547                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4548                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4549                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4550                    db 32h,20h, 20h, 20h, 0
4551                    dw 0FFFFh
4552                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4553                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4554                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4555                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4556                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4557                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4558                    db 32h,20h, 20h, 20h, 0
4559                    dw 0FFFFh
4560                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4561                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4562                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4563                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4564                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4565                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4566                    db 32h,20h, 20h, 20h, 0
4567                    dw 0FFFFh
4568                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4569                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4570                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4571                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4572                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4573                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4574                    db 32h,20h, 20h, 20h, 0
4575                    dw 0FFFFh
4576                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4577                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4578                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4579                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4580                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4581                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4582                    db 32h,20h, 20h, 20h, 0
4583                    dw 0FFFFh
4584                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4585                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4586                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4587                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4588                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4589                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4590                    db 32h,20h, 20h, 20h, 0
4591                    dw 0FFFFh
4592                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4593                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
4594                    db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
4595                    db 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
4596                    db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
4597                    db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
4598                    db 32h,20h, 20h, 20h, 0
4599                    dw 0FFFFh
4600                    db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
4601                    db 1, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
```

```
                                db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
                                db 0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
                                db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
                                db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
                                db 32h,20h, 20h, 20h, 0
                                dw 0FFFFh
                                db 0, 0, 50h, 3, 0, 2, 1, 1, 0,      2, 10h,0, 0, 0, 0F8h
                                db 1, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 3
                                db 20h,0, 28h,0, 0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0
                                db 0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0, 0, 0, 0, 0, 0,       0, 0, 0
                                db 0, 0FFh, 1, 0, 0, 0,       4Eh, 4Fh, 20h, 4Eh, 41h, 4Dh, 45h
                                db 20h,20h, 20h, 20h, 0, 0, 0,       0, 0, 46h, 41h,       54h, 31h
                                db 32h,20h, 20h, 20h, 0
                                db 0
                        %endif

                        ;--------------------------------------------------------------------------
                        ; Possibly disposable data, goes at end of data group
                        ;**************************************************************************

                        ; Possibly disposable data, goes at end of data group

                        ;***ibm_disk_io - main routine, fixes at rom bug
                        ;
                        ;   entry:  (ah) = function, 02 or 0a for read.
                        ;           (dl) = drive number (80h or 81h).
                        ;           (dh) = head number.
                        ;           (ch) = cylinder number.
                        ;           (cl) = sector number (high 2 bits has cylinder number).
                        ;           (al) = number of sectors.
                        ;           (es:bx) = address of read buffer.
                        ;           for more on register contents see rom bios listing.
                        ;           stack set up for return by an iret.
                        ;
                        ;   exit:   (ah) = status of current operation.
                        ;           (cy) = 1 if failed, 0 if successful.
                        ;           for other register contents see rom bios listing.
                        ;
                        ;   uses:
                        ;
                        ;
                        ;   warning: uses old13 vector for non-read calls.
                        ;           does direct calls to the at rom.
                        ;           does segment arithmatic.
                        ;
                        ;   effects: performs disk i/o operation.

                        ; 16/10/2022
                        ; 28/05/2019
                        cmd_block equ 42h ; ROMBIOS DATA segment (40h) offset 42h ; 13/12/2022

                        ;* offsets into cmd_block for registers

                        pre_comp equ 0     ;write pre-compensation
                        sec_cnt     equ 1  ;sector count
                        sec_num     equ 2  ;sector number
                        cyl_low     equ 3  ;cylinder number, low part
                        cyl_high equ 4     ;cylinder number, high part
                        drv_head equ 5     ;drive/head (bit 7 = ecc mode, bit 5 = 512 byte sectors,
                                           ;            bit 4 = drive number, bits 3-0 have head number)
                        cmd_reg  equ 6     ;command register

                        ; 01/10/2022
                        disk_status1       equ 74h
                        hf_num             equ 75h
                        control_byte       equ 76h

                        ibm_disk_io:
000016F2 80FA80                 cmp     dl, 80h         ; main routine,       fixes at rom bug
000016F5 720A                   jb      short atd1      ; pass through floppy disk calls.
000016F7 80FC02                 cmp     ah, 2
000016FA 740A                   jz      short atd2      ; intercept call 02 (read sectors).
000016FC 80FC0A                 cmp     ah, 0Ah
000016FF 7405                   jz      short atd2      ; and call 0Ah (read long).
                        atd1:
00001701 2EFF2E[0601]           jmp     far [cs:old13]
                                ;jmp    cs:old13        ; use rom int 13h handler
                        ;--------------------------------------------------------------------------

                        atd2:
00001706 53                     push    bx
00001707 51                     push    cx
00001708 52                     push    dx
00001709 57                     push    di
0000170A 1E                     push    ds
0000170B 06                     push    es
0000170C 50                     push    ax
0000170D B84000                 mov     ax, 40h         ; bioseg (rombios data segment)
                                                        ; establish bios segment addressing
00001710 8ED8                   mov     ds, ax
                                ; 16/10/2022
00001712 C606740000             mov     byte [disk_status1], 0
                                ;mov    byte ptr ds:74h, 0 ; [disk_status1]
                                                        ; initially no error code.
00001717 80E27F                 and     dl, 7Fh         ; mask to hard disk number
0000171A 3A167500               cmp     dl, [hf_num]
                                ;cmp    dl, ds:75h      ; [hf_num] ; 40h:75h
0000171E 7207                   jb      short atd3      ; disk number in range
                                ;mov    byte ptr ds:74h, 1 ; bad_disk
00001720 C606740001             mov     byte [disk_status1], 1
00001725 EB20                   jmp     short atd4      ; disk number out of range error,
                                                        ; return
                        ; --------------------------------------------------------------------------

                        atd3:
00001727 53                     push    bx
00001728 8CC0                   mov     ax, es
0000172A C1EB04                 shr     bx, 4           ; make es:bx to seg:000x form.
0000172D 01D8                   add     ax, bx
0000172F 8EC0                   mov     es, ax
00001731 5B                     pop     bx
00001732 83E30F                 and     bx, 0Fh
00001735 0E                     push    cs
00001736 E8DF00                 call    check_dma
00001739 720C                   jb      short atd4      ; abort if dma across segment boundary
0000173B 58                     pop     ax
0000173C 50                     push    ax
0000173D E81A00                 call    setcmd          ; set up command block for disk op
00001740 BAF603                 mov     dx, 3F6h        ; hf_reg_port
00001743 EE                     out     dx, al          ; write out command modifier
00001744 E86B00                 call    docmd           ; carry out command
                        ; --------------------------------------------------------------------------

                        atd4:
```

```
                                ;  new code - let logical or clear carry and then set carry if ah!=0
                                ;         and save a couple bytes while were at it.

00001747 58                               pop     ax
                                          ;mov    ah, ds:74h      ; [disk_status1]
00001748 8A267400                         mov     ah, [disk_status1]
0000174C 08E4                             or      ah, ah
0000174E 7401                             jz      short atd5
00001750 F9                               stc
                                atd5:
00001751 07                               pop     es
00001752 1F                               pop     ds
00001753 5F                               pop     di
00001754 5A                               pop     dx
00001755 59                               pop     cx
00001756 5B                               pop     bx
00001757 CA0200                           retf    2               ; far return, dropping flags

                                ; =============== S U B   R O U T I N E =======================================

                                ;***setcmd - set up cmd_block for the disk operation
                                ;
                                ;   entry:  (ds) = bios data segment.
                                ;           (es:bx) in seg:000x form.
                                ;           other registers as in int 13h call
                                ;
                                ;   exit:   cmd_block set up for disk read call.
                                ;           control_byte set up for disk operation.
                                ;           (al) = control byte modifier
                                ;
                                ;   sets the fields of cmd_block using the register contents
                                ;   and the contents of the disk parameter block for the given drive.
                                ;
                                ;   warning: (ax) destroyed.
                                ;           does direct calls to the at rom.

                                setcmd:                 ; proc near
                                          ;mov    ds:43h,al       ; [cmd_block+sec_cnt]
                                          ; 16/10/2022
0000175A A24300                           mov     [cmd_block+sec_cnt], al
                                          ;mov    byte ptr ds:48h, 20h ; [cmd_block+cmd_reg]
0000175D C606480020                       mov     byte [cmd_block+cmd_reg], 20h ; assume function 02h (read)
00001762 80FC02                           cmp     ah, 2
00001765 7405                             jz      short setc1     ; cmd_reg = 20h       if function 02h       (read)
00001767 C606480022                       mov     byte [cmd_block+cmd_reg], 22h
                                          ;mov    byte ptr ds:48h, 22h ; [cmd_block+cmd_reg]
                                                                  ; cmd_reg = 22h       if function 0Ah       (read long)
                                setc1:
0000176C 88C8                             mov     al, cl
0000176E 243F                             and     al, 3Fh         ; mask sector number
                                          ;mov    ds:44h,al       ; [cmd_block+sec_num]
                                          ;mov    ds:45h,ch       ; [cmd_block+cyl_low]
00001770 A24400                           mov     [cmd_block+sec_num], al ; mov [44h],al
00001773 882E4500                         mov     [cmd_block+cyl_low], ch ; mov [45h],ch
00001777 88C8                             mov     al, cl
00001779 C0E806                           shr     al, 6           ; get two high bits of cylinder      number
                                          ;mov    ds:46h,al       ; [cmd_block+cyl_high]
0000177C A24600                           mov     [cmd_block+cyl_high], al ; mov [46h],al
0000177F 89D0                             mov     ax, dx
00001781 C0E004                           shl     al, 4           ; drive number
00001784 80E40F                           and     ah, 0Fh
00001787 08E0                             or      al, ah          ; head number
00001789 0CA0                             or      al, 0A0h        ; set ecc and 512 bytes       per sector
                                          ;mov    ds:47h,al       ; [cmd_block+drv_head]
0000178B A24700                           mov     [cmd_block+drv_head], al  ; mov [47h],al
0000178E 06                               push    es
0000178F 53                               push    bx
00001790 0E                               push    cs
00001791 E85C00                           call    get_vec
00001794 268B4705                         mov     ax, [es:bx+5]  ; [es:bx+fdp_precomp]
                                                                  ; write pre-comp from disk parameters
00001798 C1E802                           shr     ax, 2
                                          ;mov    ds:42h,al       ; [cmd_block+pre_comp]
0000179B A24200                           mov     [cmd_block+pre_comp], al ; mov [42h],al
                                                                  ; only use low part
0000179E 268A4708                         mov     al, [es:bx+8]  ; [es:bx+fdp_control]
                                                                  ; control byte modifier
000017A2 5B                               pop     bx
000017A3 07                               pop     es
                                          ;mov    ah, ds:76h      ; [control_byte]
000017A4 8A267600                         mov     ah, [control_byte] ; mov ah,[76h]
000017A8 80E4C0                           and     ah, 0C0h        ; keep disable retry bits
000017AB 08C4                             or      ah, al
                                          ;mov    ds:76h,ah
000017AD 88267600                         mov     [control_byte], ah ; mov [76h],al
000017B1 C3                               retn

                                ; =============== S U B   R O U T I N E =======================================

                                ;***docmd - carry out read operation to at hard disk
                                ;
                                ;   entry:  (es:bx) = address for read in data.
                                ;           cmd_block set up for disk read.
                                ;
                                ;   exit:   buffer at (es:bx) contains data read.
                                ;           disk_status1 set to error code (0 if success).
                                ;
                                ;
                                ;
                                ;   warning: (ax), (bl), (cx), (dx), (di) destroyed.
                                ;           no check is made for dma boundary overrun.
                                ;
                                ;   effects: programs disk controller.
                                ;           performs disk input.

                                docmd:                  ; proc near
000017B2 89DF                             mov     di, bx
000017B4 0E                               push    cs
000017B5 E84000                           call    command
000017B8 7535                             jnz     short doc3
                                doc1:
000017BA 0E                               push    cs
000017BB E84200                           call    waitt           ; wait for controller to complete read
000017BE 752F                             jnz     short doc3
000017C0 B90001                           mov     cx, 256         ; 256 words per sector
000017C3 BAF001                           mov     dx, 1F0h        ; hf_port
000017C6 FC                               cld                     ; string op goes up
000017C7 FA                               cli                     ; disable interrupts
                                                                  ; (bug was forgetting this)

                                ;   M062 -- some of these old machines have intermittent failures
                                ;           when the read is done at full speed. Instead of using
                                ;           a string rep instruction, we'll use a loop. There is
```

```
4850                                    ;           a slight performance hit, but it only affects these
4851                                    ;           very old machines with an exact date code match, and
4852                                    ;           it makes said machines more reliable
4853                                    ;
4854                                    ;M062       repz    insw            ;read in sector
4855                                    
4856                                    rsct_loop:
4857 000017C8 6D                       rsct_loop:  insw
4858 000017C9 E2FD                                  loop    rsct_loop
4859 000017CB FB                                    sti
4860                                                ; 16/10/2022
4861 000017CC F606480002                            test    byte [cmd_block+cmd_reg], 02h
4862                                                ;test   byte ptr ds:48h, 2 ; [cmd_block+cmd_reg]
4863                                                                        ; (ds = 40h)
4864 000017D1 7410                                  jz      short doc2      ; no ecc bytes to read.
4865 000017D3 0E                                    push    cs
4866 000017D4 E83100                                call    wait_drq        ; wait for controller to complete read
4867 000017D7 7216                                  jb      short doc3
4868 000017D9 B90400                                mov     cx, 4           ; 4 bytes of ecc
4869 000017DC BAF001                                mov     dx, 1F0h        ; hf_port
4870 000017DF FA                                    cli
4871 000017E0 F36C                                  rep insb                ; read in ecc
4872 000017E2 FB                                    sti
4873                                    doc2:
4874 000017E3 0E                                    push    cs
4875 000017E4 E82900                                call    check_status
4876 000017E7 7506                                  jnz     short doc3      ; operation failed
4877                                                ;dec    byte ptr ds:43h         ; [cmd_block+sec_cnt]
4878 000017E9 FE0E4300                              dec     byte [cmd_block+sec_cnt]
4879 000017ED 75CB                                  jnz     short doc1      ; loop while more sectors to read
4880                                    doc3:
4881 000017EF C3                                    retn
4882                                    
4883                                    ; =============== S U B   R O U T I N E =======================================
4884                                    
4885                                    ;***define where the rom routines are actually located
4886                                    ;       in the buggy old AT BIOS that we might need to
4887                                    ;       install a special level of int13 handler for
4888                                    
4889                                    ; 16/10/2022
4890                                    
4891                                    romsegment  equ 0F000h  ; segment
4892                                    romcommand  equ 2E1Eh   ; offset in romsegment
4893                                    romwait             equ 2E7Fh   ; offset in romsegment
4894                                    romwait_drq         equ 2EE2h   ; offset in romsegment
4895                                    romcheck_status equ 2EF8h   ; offset in romsegment
4896                                    romcheck_dma        equ 2F69h   ; offset in romsegment
4897                                    romget_vec  equ 2F8Eh   ; offset in romsegment
4898                                    romfret             equ 0FF65h  ; far return in rom
4899                                    
4900                                    ;***get_vec - get pointer to hard disk parameters.
4901                                    ;
4902                                    ;   entry:  (dl) = low bit has hard disk number (0 or 1).
4903                                    ;
4904                                    ;   exit:   (es:bx) = address of disk parameters table.
4905                                    ;
4906                                    ;   uses:   ax for segment computation.
4907                                    ;
4908                                    ;   loads es:bx from interrupt table in low memory, vector 46h (disk 0)
4909                                    ;   or 70h (disk 1).
4910                                    ;
4911                                    ;   warning: (ax) destroyed.
4912                                    ;           this does a direct call to the at rom.
4913                                    
4914                                    get_vec:    ; proc near
4915                                                ;push   0FF65h          ; romfret ; far       return in rom
4916                                                ;jmp    far ptr 0F000h:2F8Eh
4917                                                ; 16/10/2022
4918 000017F0 6865FF                               push    romfret         ; far return in rom
4919 000017F3 EA8E2F00F0                           jmp     romsegment:romget_vec
4920                                    
4921                                    ; =============== S U B   R O U T I N E =======================================
4922                                    
4923                                    ;***command - send contents of cmd_block to disk controller.
4924                                    ;
4925                                    ;   entry:  control_byte
4926                                    ;           cmd_block - set up with values for hard disk controller.
4927                                    ;
4928                                    ;   exit:   disk_status1 = error code.
4929                                    ;           nz if error, zr for no error.
4930                                    ;
4931                                    ;
4932                                    ;   warning: (ax), (cx), (dx) destroyed.
4933                                    ;           does a direct call to the at rom.
4934                                    ;
4935                                    ;   effects: programs disk controller.
4936                                    
4937                                    command:    ; proc near
4938                                                ;push   0FF65h          ; romfret ; far       return in rom
4939                                                ;jmp    far ptr 0F000h:2E1Eh
4940                                                ; 16/10/2022
4941 000017F8 6865FF                               push    romfret         ; far return in rom
4942 000017FB EA1E2E00F0                           jmp     romsegment:romcommand
4943                                    
4944                                    ; =============== S U B   R O U T I N E =======================================
4945                                    
4946                                    ;***waitt - wait for disk interrupt
4947                                    ;
4948                                    ;   entry:  nothing.
4949                                    ;
4950                                    ;   exit:   disk_status1 = error code.
4951                                    ;           nz if error, zr if no error.
4952                                    ;
4953                                    ;
4954                                    ;   warning: (ax), (bl), (cx) destroyed.
4955                                    ;           does a direct call to the at rom.
4956                                    ;
4957                                    ;   effects: calls int 15h, function 9000h.
4958                                    
4959                                    waitt:              ; proc near
4960                                                ;push   0FF65h          ; romfret ; far       return in rom
4961                                                ;jmp    far ptr 0F000h:2E7Fh
4962                                                ; 16/10/2022
4963 00001800 6865FF                               push    romfret         ; far return in rom
4964 00001803 EA7F2E00F0                           jmp     romsegment:romwait
4965                                    
4966                                    ; =============== S U B   R O U T I N E =======================================
4967                                    
4968                                    ;***wait_drq - wait for data request.
4969                                    ;
4970                                    ;   entry:  nothing.
4971                                    ;
4972                                    ;   exit:   disk_status1 = error code.
4973                                    ;           cy if error, nc if no error.
```

```
                                ;
                                ;   warning: (al), (cx), (dx) destroyed.
                                ;           does a direct call to the at rom.

                                wait_drq:   ; proc near
                                            ;push   0FF65h          ; romfret ; far       return in rom
                                            ;jmp    far ptr 0F000h:2EE2h
                                            ; 16/10/2022
00001808 6865FF                             push    romfret         ; far return in rom
0000180B EAE22E00F0                         jmp     romsegment:romwait_drq

                                ; ============== S U B   R O U T I N E =======================================

                                ;***check_status - check hard disk status.
                                ;
                                ;   entry:  nothing.
                                ;
                                ;   exit:   disk_status1 = error code.
                                ;           nz if error, zr if no error.
                                ;
                                ;   warning: (ax), (cx), (dx) destroyed.
                                ;           does a direct call to the at rom.

                                check_status:       ; proc near
                                            ;push   0FF65h          ; romfret ; far       return in rom
                                            ;jmp    far ptr 0F000h:2EF8h
                                            ; 16/10/2022
00001810 6865FF                             push    romfret         ; far return in rom
00001813 EAF82E00F0                         jmp     romsegment:romcheck_status

                                ; ============== S U B   R O U T I N E =======================================

                                ;***check_dma - check for dma overrun 64k segment.
                                ;
                                ;   entry:  (es:bx) = addr. of memory buffer in seg:000x form.
                                ;           cmd_block set up for operation.
                                ;
                                ;   exit:   disk_status1 - error code.
                                ;           cy if error, nc if no error.
                                ;
                                ;   warning: does a direct call to the at rom.

                                check_dma:  ; proc near
                                            ;push   0FF65h          ; romfret ; far       return in rom
                                            ;jmp    far ptr 0F000h:2F69h
                                            ; 16/10/2022
00001818 6865FF                             push    romfret         ; far return in rom
0000181B EA692F00F0                         jmp     romsegment:romcheck_dma

                                ;--------------------------------------------------------------------------

                                endatrom:

                                ; ------------------------------------------------------------------------

                                ;; M015 -- begin changes
                                ;;
                                ;; Certain old COMPAQ '286 machines have a bug in their ROM BIOS.
                                ;; When Int13 is done with AH > 15h and DL >= 80h, they trash
                                ;; the byte at DS:74h, assuming that DS points to ROM_DATA.
                                ;; If our init code detects this error, it will install this
                                ;; special Int13 hook through the same mechanism that was set
                                ;; up for the IBM patch above. This code is also dynamically
                                ;; relocated by MSINIT.

                                compaq_disk_io:
00001820 80FC15                             cmp     ah, 15h         ; compaq_disk_io proc far
                                                                    ;
                                                                    ; the following       label defines the end of the at      rom patch.
                                                                    ; this is used at configuration      time.
                                                                    ;
                                                                    ; warning!!!
                                                                    ; this code will be dynamically      relocated by msinit
00001823 7705                               ja      short mebbe_hookit ; only deal with functions > 15h
                                no_hookit:
                                            ;jmp    cs:Old13
                                            ; 16/10/2022
00001825 2EFF2E[0601]                       jmp     far [cs:Old13]

                                ; ------------------------------------------------------------------------

                                mebbe_hookit:
0000182A 80FA80                             cmp     dl, 80h
0000182D 72F6                               jb      short no_hookit
0000182F 1E                                 push    ds

                                            ; 07/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                            ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:1830h
                                            ;push   ax
                                            ;mov    ax, 40h
                                            ;mov    ds, ax
                                            ;pop    ax
00001830 6A40                               push    40h
00001832 1F                                 pop     ds

00001833 9C                                 pushf
                                            ;call   cs:Old13
                                            ; 16/10/2022
00001834 2EFF1E[0601]                       call    far [cs:Old13]
00001839 1F                                 pop     ds
0000183A CA0200                             retf    2

                                ; ------------------------------------------------------------------------

0000183D 00                     end_compaq_i13hook: db 0

                                ; ============== S U B   R O U T I N E =======================================

                                ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                %if 0

                                ; CMOS Clock setting support routines used by MSCLOCK.
                                ; Warning!!! This code will be dynamically relocated by MSINIT.

                                daycnt_to_day:      ; proc far

                                ; entry: [daycnt] = number of days since 1-1-80
                                ;
                                ; return: ch - century in bcd
                                ;      cl - year in bcd
                                ;      dh - month in bcd
                                ;      dl - day in bcd

                                            ; 16/10/2022
```

```
                                push    word [cs:daycnt] ; save daycnt
                                cmp     word [cs:daycnt], 7305; (365*20+(20/4))
                                                        ; # days from 1-1-1980 to 1-1-2000
                                jnb     short century20
                                mov     byte [cs:base_century], 19
                                mov     byte [cs:base_year], 80
                                jmp     short years
                        ; ---------------------------------------------------------------------------

                        century20:
                                mov     byte [cs:base_century], 20
                                mov     byte [cs:base_year], 0
                                sub     word [cs:daycnt], 7305; (365*20+(20/4))
                                                        ; adjust daycnt
                        years:
                                xor     dx, dx
                                mov     ax, [cs:daycnt]
                                mov     bx, 1461        ; (366+365*3)
                                                        ; # of days in a Leap year block
                                div     bx              ; AX = # of leap block,       DX = daycnt
                                mov     [cs:daycnt], dx       ; save daycnt left
                                mov     bl, 4
                                mul     bl              ; AX = # of years. Less       than 100
                                add     [cs:base_year], al ; So, ah = 0. Adjust year
                                inc     word [cs:daycnt]      ; set daycnt to       1 base
                                cmp     word [cs:daycnt], 366 ; daycnt=remainder of leap year      bk
                                jbe     short leapyear ; within 366+355+355+355 days.
                                inc     byte [cs:base_year]   ; if daycnt <= 366, then leap year
                                sub     word [cs:daycnt], 366 ; else daycnt--, base_year++ ;
                                mov     cx, 3           ; And next three years are normal
                        regularyear:
                                cmp     word [cs:daycnt], 365 ; for(i=1; i>3 or daycnt <=365;      i++)
                                jbe     short yeardone ; {if (daycnt >       365)
                                inc     byte [cs:base_year]   ;   { daycnt -=       365
                                sub     word [cs:daycnt], 365 ;   }
                                loop    regularyear     ; }
                                                        ;
                                                        ; should never fall through loop
                        leapyear:
                                mov     byte [cs:month_tab+1], 29 ; leap year.
                                                        ; change month table.
                        yeardone:
                                xor     bx, bx
                                xor     dx, dx
                                mov     ax, [cs:daycnt]
                                ;mov    si, offset month_tab
                                mov     si, month_tab  ; 19/10/2022
                                mov     cx, 12
                        months:
                                inc     bl

                                ; !!! -- 16/10/2022 -- (if DS=CS, what for CS: prefixes are used !?)
                                ;mov    dl, [cs:si]
                                ; !!! -- 16/10/2022 -- (may be to keep code addrs as unchanged/fix!?)
                                ; ds = cs !? ((ofcourse ds must be same with cs here))
                                ;mov    dl, [si] ; 20/03/2019 (MSDOS 6.21 IO.SYS, BIOSDATA:14C0h)
                                ;mov    dl, [si] ; 16/10/2022 (MSDOS 5.0 IO.SYS, BIOSDATA:14C0h)

                                mov     dl, [si] ; ?    ; mov dl, [cs:si]
                                cmp     ax, dx          ; cmp daycnt for each month till fit
                                                        ; dh=0
                                jbe     short month_done
                                inc     si              ; next month
                                sub     ax, dx          ; adjust daycnt
                                loop    months          ;
                                                        ; should never fall through loop
                        month_done:
                                mov     byte [cs:month_tab+1], 28
                                                        ; restore month table value
                                mov     dl, bl
                                mov     dh, [cs:base_year]
                                mov     cl, [cs:base_century] ; al=day,dl=month,dh=year,cl=cntry
                                call    far [cs:bintobcd]
                                ;call   cs:bintobcd     ; convert "day"       to bcd
                                                        ; dl = bcd day,       al = month
                                xchg    dl, al
                                call    far [cs:bintobcd]
                                ;call   cs:bintobcd     ; dh = bcd month, al = year
                                xchg    dh, al
                                call    far [cs:bintobcd]
                                ;call   cs:bintobcd     ; cl = bcd year, al = century
                                xchg    cl, al
                                call    far [cs:bintobcd]
                                ;call   cs:bintobcd     ; ch = bcd century
                                mov     ch, al
                                pop     word [cs:daycnt] ; restore original value
                                retf

                        enddaycnttoday:

                        %endif

                        ; ============== S U B   R O U T I N E =======================================

                        ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                        %if 0

                        bin_to_bcd: ; proc far            ; real time clock support

                        ;convert a binary input in al (less than 63h or 99 decimal)
                        ;into a bcd value in al. ah destroyed.

                                    push    cx
                                    aam                 ; al=high digit       bcd, ah=low digit bcd
                                    mov     cl, 4
                                    shl     ah, cl      ; mov the high digit to       high nibble
                                    or      al, ah
                                    pop     cx
                                    retf
                        %endif

                        ; ---------------------------------------------------------------------------

                        ; the k09 requires the routines for reading the clock because of the suspend/
                        ; resume facility. the system clock needs to be reset after resume.

                        ; the following routine is executed at resume time when the system
                        ; powered on after suspension. it reads the real time clock and
                        ; resets the system time and date, and then irets.

                        ; warning!!! this code will be dynamically relocated by msinit.

                            ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                            ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:183Eh
```

```
                                int_6Ch:
0000183E 0E                             push    cs
0000183F 1F                             pop     ds
                                        ;cmp    byte [cs:inHMA], 0
00001840 803E[0D00]00                   cmp     byte [inHMA], 0
00001845 7405                           jz      short int6c
00001847 BB[2A07]                       mov     bx, EnsureA20On
0000184A FFD3                           call    bx
                                int6c:
                                        ;push   cs
                                        ;pop    ds
0000184C 8F06[F805]                     pop     word [int6c_ret_addr] ; pop off return address
00001850 8F06[FA05]                     pop     word [int6c_ret_addr+2]
00001854 9D                             popf
00001855 E81300                         call    read_real_date ; get the date from the clock
00001858 FA                             cli
00001859 8936[8904]                     mov     [daycnt], si   ; update dos copy of date
0000185D FB                             sti
0000185E E8B900                         call    read_real_time ; get the time from the      rtc
00001861 FA                             cli
00001862 B401                           mov     ah, 1
00001864 CD1A                           int     1Ah             ; CLOCK - SET TIME OF DAY
                                                                ; CX:DX = clock count
                                                                ; Return: time of day set
00001866 FB                             sti
                                        ;jmp    int6c_ret_addr ; long jump
                                        ; 16/10/2022
00001867 FF2E[F805]                     jmp     far [int6c_ret_addr] ; long jump

                                ; =============== S U B   R O U T I N E =======================================

                                ;   read_real_date reads real-time clock for date and returns the number
                                ;   of days elapsed since 1-1-80 in si

                                read_real_date:     ; proc near
0000186B 50                             push    ax
0000186C 51                             push    cx
0000186D 52                             push    dx
0000186E 30E4                           xor     ah, ah          ; throw away clock roll       over
00001870 CD1A                           int     1Ah             ; CLOCK - GET TIME OF DAY
                                                                ; Return: CX:DX       = clock count
                                                                ; AL = 00h if clock was       read or written (via AH=0,1) since the previous
                                                                ; midnight
                                                                ; Otherwise, AL       > 0
00001872 5A                             pop     dx
00001873 59                             pop     cx
00001874 58                             pop     ax
00001875 50                             push    ax
00001876 53                             push    bx
00001877 51                             push    cx
00001878 52                             push    dx
                                        ;mov    word [cs:daycnt2], 1
                                        ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                        ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:187Ah
00001879 C706[0006]0100                 mov     word [daycnt2], 1
                                                                ; REAL TIME CLOCK ERROR       FLAG (+1 DAY)
0000187F B404                           mov     ah, 4
00001881 CD1A                           int     1Ah             ; CLOCK - READ DATE FROM REAL TIME CLOCK (AT,XT286,CONV,PS)
                                                                ; Return: DL = day in BCD
                                                                ; DH = month in       BCD
                                                                ; CL = year in BCD
                                                                ; CH = century (19h or 20h)
00001883 7303                           jnb     short read_ok
00001885 E98300                         jmp     r_d_ret
                                ;-------------------------------------------------------------------------------

                                read_ok:
00001888 882E[FC05]                     mov     [bin_date_time], ch
0000188C 880E[FD05]                     mov     [bin_date_time+1], cl
00001890 8836[FE05]                     mov     [bin_date_time+2], dh
00001894 8816[FF05]                     mov     [bin_date_time+3], dl
                                        ;mov    word [cs:daycnt2], 2 ; READ OF R-T CLOCK SUCCESSFUL
                                        ; 08/08/2023
                                        ;mov    byte [daycnt2], 2
00001898 FE06[0006]                     inc     byte [daycnt2] ; 2
0000189C E83401                         call    bcd_verify     ; verify bcd values in range
0000189F 726A                           jb      short r_d_ret  ; some value out of range
                                        ;mov    word [cs:daycnt2], 3
                                        ; 08/08/2023
                                        ;mov    byte [daycnt2], 3
000018A1 FE06[0006]                     inc     byte [daycnt2] ; 3
000018A5 E8DB00                         call    date_verify
000018A8 7261                           jb      short r_d_ret
                                        ;mov    word [cs:daycnt2], 0
                                        ; 08/08/2023
000018AA C606[0006]00                   mov     byte [daycnt2], 0
000018AF E8A100                         call    in_bin
000018B2 A0[FD05]                       mov     al, [bin_date_time+1]
000018B5 98                             cbw
000018B6 803E[FC05]14                   cmp     byte [bin_date_time], 20 ; 20th century?
000018BB 7503                           jnz     short century_19 ; no
000018BD 83C064                         add     ax, 100         ; add in a century
                                century_19:
000018C0 83E850                         sub     ax, 80          ; subtract off 1-1-80
000018C3 B104                           mov     cl, 4           ; leap year every 4
000018C5 F6F1                           div     cl              ; al= # leap year blocks, ah= remainder
000018C7 88E3                           mov     bl, ah          ; save odd years
000018C9 98                             cbw                     ; zero ah
000018CA B9B505                         mov     cx, 1461        ; 366+(3*365)
                                                                ; # of days in leap year blocks
000018CD F7E1                           mul     cx
                                        ;mov    [cs:daycnt2], ax ; SAVE COUNT OF DAYS
                                        ; 08/08/2023
000018CF A3[0006]                       mov     [daycnt2], ax
000018D2 88D8                           mov     al, bl          ; get odd years       count
000018D4 98                             cbw
000018D5 09C0                           or      ax, ax
000018D7 740B                           jz      short leap_year
000018D9 B96D01                         mov     cx, 365         ; days in year
000018DC F7E1                           mul     cx
                                        ;add    [cs:daycnt2], ax ; ADD ON DAYS IN ODD YEARS
                                        ; 08/08/2023
000018DE 0106[0006]                     add     [daycnt2], ax
000018E2 EB07                           jmp     short leap_adjustment ;       account for leap year
                                                                ; possibly account for a leap day
                                ;-------------------------------------------------------------------------------

                                leap_year:
000018E4 803E[FE05]02                   cmp     byte [bin_date_time+2], 2 ; is       month february?
000018E9 7604                           jbe     short no_leap_adjustment ; jan or feb. no leap day yet.
                                leap_adjustment:
                                        ;inc    word [cs:daycnt2] ; account for leap day
                                        ; 08/08/2023
000018EB FF06[0006]                     inc     word [daycnt2]
```

```
                                no_leap_adjustment:
000018EF 8A0E[FF05]                     mov     cl, [bin_date_time+3] ; get days of month
000018F3 30ED                           xor     ch, ch
000018F5 49                             dec     cx              ; because of offset from day 1,      not day 0
                                        ;add    [cs:daycnt2], cx ; GET DAYS IN MONTHS PRECEEDING
                                        ; 08/08/2023
000018F6 010E[0006]                     add     [daycnt2], cx
000018FA 8A0E[FE05]                     mov     cl, [bin_date_time+2] ; get month
                                        ; 08/08/2023
                                        ;xor    ch, ch
000018FE 49                             dec     cx              ; january starts at offset 0

                                        ; 08/08/2023
                                        ;shl    cx, 1           ; word offset
                                        ;;mov   si, month_table
                                        ;add    si, cx
                                        ;; 16/10/2022
                                        ;; ds must be same with cs here, if so..
                                        ;; what for cs: prefixes are used !?)
                                        ;; mov  ax, [cs:si]
                                        ;; mov  ax, [si] ; 16/10/2022 (MSDOS 5.0 IO.SYS - BIOSDATA:15D5h)
                                        ;mov    ax, [si]        ; mov ax, [cs:si]
                                        ;                       ; get # days in previous months
                                        ;add    [cs:daycnt2], ax

                                        ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                        ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:1907h
000018FF B400                           mov     ah, 0
00001901 BE[8F04]                       mov     si, month_tab
                                r_d_sum_loop:
00001904 AC                             lodsb
00001905 0106[0006]                     add     [daycnt2], ax
00001909 E2F9                           loop    r_d_sum_loop
                                r_d_ret:
                                        ;mov    si, [cs:daycnt2]
                                        ; 08/08/2023
0000190B 8B36[0006]                     mov     si, [daycnt2]
0000190F 5A                             pop     dx
00001910 59                             pop     cx
00001911 5B                             pop     bx
00001912 58                             pop     ax
00001913 C3                             retn

                                ;--------------------------------------------------------------------------

                                r_t_retj:
00001914 31C9                           xor     cx, cx
00001916 31D2                           xor     dx, dx
00001918 EB38                           jmp     short r_t_ret

                                ; =============== S U B   R O U T I N E =======================================

                                ; read_real_time reads the time from the rtc. on exit, it has the number of
                                ; ticks (at 18.2 ticks per sec.) in cx:dx.

                                read_real_time:     ; proc near
0000191A B402                           mov     ah, 2
0000191C CD1A                           int     1Ah             ; CLOCK - READ REAL TIME CLOCK (AT,XT286,CONV,PS)
                                                                ; Return: CH = hours in       BCD
                                                                ; CL = minutes in BCD
                                                                ; DH = seconds in BCD
0000191E 72F4                           jb      short r_t_retj
00001920 882E[FC05]                     mov     [bin_date_time], ch ; hours
00001924 880E[FD05]                     mov     [bin_date_time+1], cl ; minutes
00001928 8836[FE05]                     mov     [bin_date_time+2], dh ; seconds
0000192C C606[FF05]00                   mov     byte [bin_date_time+3], 0 ; unused for time
00001931 E89F00                         call    bcd_verify
00001934 72DE                           jb      short r_t_retj
00001936 E88500                         call    time_verify
00001939 72D9                           jb      short r_t_retj
0000193B E81500                         call    in_bin          ; from bcd to bin
0000193E 8A2E[FC05]                     mov     ch, [bin_date_time]
00001942 8A0E[FD05]                     mov     cl, [bin_date_time+1]
00001946 8A36[FE05]                     mov     dh, [bin_date_time+2]
0000194A 8A16[FF05]                     mov     dl, [bin_date_time+3]
                                        ; 16/10/2022
                                        ; 17/09/2022
                                        ; 31/05/2019
0000194E FF1E[0606]                     call    far [ttticks]
                                        ;call   dword ptr ttticks ; note: indirect far call
                                                                ; cx:dx = number of ticks
                                                                ; (at 18.2 ticks per sec.)
                                r_t_ret:
00001952 C3                             retn

                                ; =============== S U B   R O U T I N E =======================================

                                ;   in_bin converts bin_date_time values from bcd to bin

                                in_bin:             ; proc near
00001953 A0[FC05]                       mov     al, [bin_date_time] ; century or hours
00001956 E81F00                         call    bcd_to_bin
00001959 A2[FC05]                       mov     [bin_date_time], al
0000195C A0[FD05]                       mov     al, [bin_date_time+1] ; years or minutes
0000195F E81600                         call    bcd_to_bin
00001962 A2[FD05]                       mov     [bin_date_time+1], al
00001965 A0[FE05]                       mov     al, [bin_date_time+2] ; months or seconds
00001968 E80D00                         call    bcd_to_bin
0000196B A2[FE05]                       mov     [bin_date_time+2], al
0000196E A0[FF05]                       mov     al, [bin_date_time+3] ; days (not used for time)
00001971 E80400                         call    bcd_to_bin
00001974 A2[FF05]                       mov     [bin_date_time+3], al
00001977 C3                             retn

                                ; =============== S U B   R O U T I N E =======================================

                                ;   bcd_to_bin converts two bcd nibbles in al (value <= 99.) to
                                ;   a binary representation in al
                                ;   ah is destroyed

                                bcd_to_bin: ; proc near
00001978 88C4                           mov     ah, al
0000197A 240F                           and     al, 0Fh
0000197C B104                           mov     cl, 4
0000197E D2EC                           shr     ah, cl
00001980 D50A                           aad
00001982 C3                             retn

                                ; =============== S U B   R O U T I N E =======================================

                                ;   date_verify loosely checks bcd date values to be in range
                                ;   in bin_date_time

                                date_verify:        ; proc near
```

```
5470 00001983 803E[FC05]20                      cmp     byte [bin_date_time], 20h ; century check
5471 00001988 7732                              ja      short date_error
5472 0000198A 740E                              jz      short century_20 ; jmp in 21th century
5473 0000198C 803E[FC05]19                      cmp     byte [bin_date_time], 19h ; century check
5474                                            ;jb     short date_error
5475                                            ; 12/12/2022
5476 00001991 722A                              jb      short date_err2
5477 00001993 803E[FD05]80                      cmp     byte [bin_date_time+1], 80h ; year check
5478                                            ;jb     short date_error
5479                                            ; 12/12/2022
5480 00001998 7223                              jb      short date_err2
5481                                    century_20:
5482 0000199A 803E[FD05]99                      cmp     byte [bin_date_time+1], 99h ; year check
5483 0000199F 771B                              ja      short date_error
5484 000019A1 803E[FE05]12                      cmp     byte [bin_date_time+2], 12h ; month check
5485 000019A6 7714                              ja      short date_error
5486 000019A8 803E[FE05]00                      cmp     byte [bin_date_time+2], 0
5487                                            ;jbe    short date_error
5488 000019AD 760D                              jna     short date_error
5489 000019AF 803E[FF05]31                      cmp     byte [bin_date_time+3], 31h ; day check
5490 000019B4 7706                              ja      short date_error
5491                                            ;cmp    byte [bin_date_time+3], 0 ; day check
5492                                            ;;jbe   short date_error
5493                                            ;jna    short date_error
5494                                            ; 12/12/2022
5495                                            ; cf=0
5496                                            ;clc
5497                                            ; 12/12/2022
5498 000019B6 803E[FF05]01                      cmp     byte [bin_date_time+3], 1 ; day check
5499 000019BB C3                                retn
5500                                    ;---------------------------------------------------------------------------
5501
5502                                    date_error:
5503 000019BC F9                                stc
5504                                    date_err2:
5505 000019BD C3                                retn
5506
5507                                    ; =============== S U B   R O U TI N E =======================================
5508
5509                                    ; time_verify very loosely checks bcd date values to be in range
5510                                    ; in bin_date_time
5511
5512                                    time_verify:        ; proc near
5513 000019BE 803E[FC05]24                      cmp     byte [bin_date_time], 24h ; hour check
5514 000019C3 770C                              ja      short time_error
5515 000019C5 803E[FD05]59                      cmp     byte [bin_date_time+1], 59h ; minute check
5516 000019CA 7705                              ja      short time_error
5517                                            ; 12/12/2022h
5518                                            ;cmp    byte [bin_date_time+2], 59h ; second check
5519                                            ;ja     short time_error
5520                                            ;clc
5521                                            ;retn
5522                                            ; 12/12/2022
5523 000019CC 803E[FE05]5A                      cmp     byte  [bin_date_time+2], 5Ah
5524                                    time_error:
5525                                    bv_error:
5526 000019D1 F5                                cmc     ; cf=0 -> cf=1, cf=1 -> cf=0
5527 000019D2 C3                                retn
5528
5529                                    ; ---------------------------------------------------------------------------
5530
5531                                    ;time_error:
5532                                            ;stc
5533                                            ;retn
5534
5535                                    ; =============== S U B   R O U TI N E =======================================
5536
5537                                    ;   bcd_verify checks values in bin_date_time to be valid
5538                                    ;   bcd numerals.  carry set if any nibble out of range
5539
5540                                    bcd_verify: ; proc near
5541 000019D3 B90400                            mov     cx, 4           ; 4 bytes to check
5542 000019D6 BB[FC05]                          mov     bx, bin_date_time
5543                                    bv_loop:
5544 000019D9 8A07                              mov     al, [bx]        ; get a bcd number (0..99)
5545 000019DB 88C4                              mov     ah, al
5546 000019DD 250FF0                            and     ax, 0F00Fh      ; 10's place in high ah, 1's in al
5547                                                                    ; is 1's place in range?
5548 000019E0 3C0A                              cmp     al, 10
5549 000019E2 77ED                              ja      short bv_error ; jmp out of range
5550 000019E4 D0EC                              shr     ah, 1
5551 000019E6 D0EC                              shr     ah, 1
5552 000019E8 D0EC                              shr     ah, 1
5553 000019EA D0EC                              shr     ah, 1
5554 000019EC 80E40F                            and     ah, 0Fh         ; get rid of any erroneous bits
5555 000019EF 80FC0A                            cmp     ah, 10          ; is 10's place in range
5556 000019F2 77DD                              ja      short bv_error ; jmp out of range
5557 000019F4 43                                inc     bx              ; next byte
5558 000019F5 49                                dec     cx
5559 000019F6 75E1                              jnz     short bv_loop
5560 000019F8 F8                                clc                     ; set success flag
5561 000019F9 C3                                retn
5562                                    ; ---------------------------------------------------------------------------
5563
5564                                            ; 12/12/2022
5565                                    ;bv_error:
5566                                            ;stc                    ; set error flag
5567                                            ;retn
5568
5569                                    ; ---------------------------------------------------------------------------
5570
5571                                    endk09:
5572
5573                                    ; ---------------------------------------------------------------------------
5574
5575                                    ;---------------------------------------------------------------------------
5576                                    ;                                                                       :
5577                                    ;   System initialization                                              :
5578                                    ;                                                                       :
5579                                    ;   The entry conditions are established by the bootstrap                      :
5580                                    ;   loader and are considered unknown. The following jobs                       :
5581                                    ;   will be performed by this module:                                   :
5582                                    ;                                                                       :
5583                                    ;   1.      All device initialization is performed                          :
5584                                    ;   2.      A local stack is set up and DS:SI are set                   :
5585                                    ;           to point to an initialization table. Then                   :
5586                                    ;           an inter-segment call is made to the first                  :
5587                                    ;           byte of the dos                                             :
5588                                    ;   3.      Once the dos returns from this call the ds                  :
5589                                    ;           register has been set up to point to the start                  :
5590                                    ;           of free memory. The initialization will then                :
5591                                    ;           load the command program into this area                     :
5592                                    ;           beginning at 100 hex and transfer control to                :
5593                                    ;           this program.                                               :
```

```
                                ;                                                                       :
                                ;-------------------------------------------------------------------------

                                ; 01/10/2022
                                ; 08/01/2018 - Retro DOS v4.0

                                ; drvfat must be the first location of freeable space!

                                align 2
                                        ;db 90h

                                ; 09/12/2023 - Retro DOS v5.0 IO.SYS (IBMBIO.COM 7.1)
                                ; (PCDOS 7.1 IBMBIO.COM, BIOSDATA:1A0Ch)

                                ; 30/12/2022
                                ; (MSDOS 6.21 IO.SYS, BIOSDATA:16D6h)

000019FA 0000                   drvfat:         dw 0                    ; drive and fat id of dos
                                ; 09/12/2023
                                ;bios_l:    dw 0                ; first sector of data (low word)
                                ;bios_h:    dw 0                ; first sector of data (high word)
                                First_Data_Sector:
000019FC 0000                           dw 0
000019FE 0000                           dw 0
00001A00 0000                   doscnt:         dw 0                    ; how many sectors to read
                                ;fbigfat:   db 0                ; flags for drive
00001A02 0000                   fatloc:         dw 0                    ; seg addr of fat sector
00001A04 0000                   init_bootseg:   dw 0                    ; seg addr of buffer for reading boot record
                                ; 09/12/2023
00001A06 00                     fbigfat:    db 0                ; flags for drive
00001A07 80                     rom_drv_num:    db 80h                  ; rom drive number
00001A08 0002                   md_sectorsize:  dw 200h                 ; used by get_fat_sector proc.
                                ; 12/12/2023
                                ;temp_cluster:  dw 0                    ; used by get_fat_sector proc.
00001A0A FFFF                   last_fat_sec_num: dw 0FFFFh             ; used by get_fat_sector proc.

                                ; the following two bytes are used to save the info returned by int 13, ah = 8
                                ; call to determine drive parameters.

00001A0C 02                     num_heads:  db 2    ; dw 2          ; number of heads returned by rom
00001A0D 00                                 db 0    ; 09/12/2023
                                ;sec_trk:   db 9                    ; sec/trk returned by rom
00001A0E 28                     num_cyln:   db 40   ; dw 40         ; number of cylinders returned by rom
00001A0F 00                                 db 0    ; 09/12/2023
                                ; 09/12/2023
00001A10 09                     sec_trk:    db 9                    ; sec/trk returned by rom
00001A11 00                     fakefloppydrv:      db 0                    ; if 1, then no diskette drives       in the system.

                                ; 09/12/2023 - Retro DOS v5.0 IO.SYS/IBMBIO.COM
                                Orig_Int1Eh_Table:
00001A12 0000                           dw 0
00001A14 0000                           dw 0

                                ; ---------------------------------------------------------------------------

                                ; 09/12/2023
                                %if 0

                                disktable:  dw 512,0100h, 64, 0   ; warning !!! old values
                                            dw 2048, 0201h, 112, 0
                                            dw 8192, 0402h, 256, 0
                                            dw 32680, 0803h, 512, 0       ; warning !!! old values
                                            dw 65535, 1004h, 1024, 0
                                                                    ; default disktable under
                                                                    ; the assumption of total fat size <= 128 kb,
                                                                    ; and the maximum size of fat entry = 16 bit.
                                %endif

                                ; 09/12/2023 - Retro DOS v5.0 IO.SYS (IBMBIO.COM 7.1)
                                ; (PCDOS 7.1 IBMBIO.COM, BIOSDATA:1A2Ah)

                                            ; 09/12/2023
                                            ; 08/08/2023
                                            ; disktable.totalsectors: resw 1 ; high word
                                            ;                       resw 1 ; low word
                                            ; disktable.shiftcount:   resb 1
                                            ; disktable.secperclus:   resb 1
                                            ; disktable.rdirentries:  resw 1
                                            ; disktable.bigflag:      resw 1
00001A16 0000A87F0308000200-    disktable2: dw 0, 32680, 0803h, 512, 0 ; for compatibility.
00001A1F 00
                                                                    ; (32680 sectors, 16340 KB)
00001A20 040000000204000240-                dw 4, 0, 0402h, 512, 40h   ; covers upto 134 mb media.
00001A29 00
                                                                    ; fbig = 40h  ; (40000h sectors = 128 MB)
00001A2A 080000000308000240-                dw 8, 0, 0803h, 512, 40h   ; upto 268 mb ; (80000h sectors = 256 MB)
00001A33 00
00001A34 100000000410000240-                dw 16, 0, 1004h, 512, 40h  ; upto 536 mb ; (100000h sectors = 512 MB)
00001A3D 00
00001A3E 200000000520000240-                dw 32, 0, 2005h, 512, 40h  ; upto 1072 mb ; (200000h sectors = 1024 MB)
00001A47 00
00001A48 400000000640000240-                dw 64, 0, 4006h, 512, 40h  ; upto 2144 mb ; (400000h sectors = 2048 MB)
00001A51 00
                                            ; 09/12/2023
                                            ;dw 128, 0, 8007h, 512, 40h ; upto 4288 mb ; (800000h sectors = 4096 MB)
00001A52 FFFFFFFF0308000060-                dw 0FFFFh, 0FFFFh, 0803h, 0, 60h ; FAT32 (> 2144MB)
00001A5B 00
                                                                    ; (fbig and fbigbig flags are set)

                                ; ---------------------------------------------------------------------------

                                ;*******************************************************
                                ;variables for mini disk initialization
                                ;*******************************************************

                                ; 01/10/2022
                                ; [ Note: Minidisk == logical dos drive (in extended dos partition) ]

00001A5C 00                     rom_minidisk_num: db 0                  ; temp variable       for phys unit
00001A5D 00                     hnum:               db 0                ; real number of hardfiles
00001A5E [3C05]                 last_dskdrv_table: dw dskdrvs           ; index into dskdrv table
00001A60 [4C08]                 end_of_bdss:        dw bdss             ; offset value of the ending address
                                                                    ; of bds table. needed to figure out
                                                                    ; the dosdatasg address.
00001A62 0000                   mini_hdlim: dw 0
00001A64 0000                   mini_seclim:        dw 0

                                ; 19/12/2023
                                ; 09/12/2023
                                ; (PCDOS 7.1 IBMBIO.COM, BIOSDATA:1A7Ah)
                                ;ld_p_number:       dw 2BADh            ; (for 'find_mini_partition' proc)

                                ;end of mini disk init variables **********************
```

```
                                ; ----------------------------------------------------------------------------
00001A66 30312F31302F383400     bios_date:  db '01/10/84',0       ; used for checking at rom bios      date.

                                ; 13/12/2022
                                %if 0

                                ;align 2
                                            db  90h

                                ; the following are the recommended bpbs for the media that we know of so far.

                                ;struc bpbx
                                ;   resw 1 ; 512
                                ;   resb 1
                                ;   resw 1 ; 1
                                ;   resb 1 ; 2
                                ;   resw 1
                                ;   resw 1
                                ;   resb 1
                                ;   resw 1
                                ;   resw 1
                                ;   resw 1 ; 2
                                ;   resw 1
                                ;   resw 1 ; hidden sector high
                                ;   resd 1 ; extended total sectors
                                ;.size:
                                ;endstruc

                                ; 08/01/2019 - Retro DOS v4.0

                                ; 20/04/2019

                                ; 01/10/2022 - Retro DOS v4.0 (MSDOS 5.0) IO.SYS

                                ; 09/12/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
                                ; (PCDOS 7.1 IBMBIO.COM, BIOSDATA:1A86h)

                                ; 09/12/2022
                                BPB48T:
                                ;bpb48t:    ; bpbx <512, 2, 1, 2, 112, 720, 0FDh, 2, 9, 2, 0, 0, 0, 0>
                                            ; 48 tpi diskettes    ;
                                            dw      512             ; physical sector size in bytes
                                            db      2               ; sectors/allocation unit
                                            dw      1               ; reserved sectors for dos
                                            db      2               ; number of allocation tables
                                            dw      112             ; number of directory entries
                                            dw      720 ; 2*9*40    ; number of sectors (at 512 bytes each)
                                            db      0FDh            ; media descriptor
                                            dw      2               ; number of fat sectors
                                            dw      9               ; sectors per track
                                            dw      2               ; heads
                                            dw      0               ; hidden sector count (low word)
                                            dw      0               ; hidden sector (high)
                                            dw      0               ; number of sectors (low)
                                            dw      0               ; number of sectors (high)
                                            ; 09/12/2023
                                            ; FAT32 extensions (to BDS)
                                            times   28 db 0
                                            ;
                                            db 90h
                                ;align 2
                                BPB96T:
                                ;bpb96t:    ; bpbx <512, 1, 1, 2, 224, 2400, 0F9h, 7, 15, 2, 0, 0, 0, 0>
                                            ; 96 tpi diskettes    ;
                                            dw      512             ; physical sector size in bytes
                                            db      1               ; sectors/allocation unit
                                            dw      1               ; reserved sectors for dos
                                            db      2               ; number of allocation tables
                                            dw      224             ; number of directory entries
                                            dw      2400 ; 2*15*80  ; number of sectors (at 512 bytes each)
                                            db      0F9h            ; media descriptor
                                            dw      7               ; number of fat sectors
                                            dw      15              ; sectors per track
                                            dw      2               ; heads
                                            dw      0               ; hidden sector count (low word)
                                            dw      0               ; hidden sector (high)
                                            dw      0               ; number of sectors (low)
                                            dw      0               ; number of sectors (high)
                                            ; 09/12/2023
                                            ; FAT32 extensions (to BDS)
                                            times   28 db 0
                                            ;
                                            db 90h
                                ;align 2
                                BPB35:
                                ;bpb35:             ; bpbx <512, 2, 1, 2, 112, 1440, 0F9h, 3, 9, 2, 0, 0, 0, 0>
                                            ; 3.5" diskettes - 720 KB ;
                                            dw      512             ; physical sector size in bytes
                                            db      2               ; sectors/allocation unit
                                            dw      1               ; reserved sectors for dos
                                            db      2               ; number of allocation tables
                                            dw      112             ; number of directory entries
                                            dw      1440 ; 2*9*80   ; number of sectors (at 512 bytes each)
                                            db      0F9h            ; media descriptor
                                            dw      3               ; number of fat sectors
                                            dw      9               ; sectors per track
                                            dw      2               ; heads
                                            dw      0               ; hidden sector count (low word)
                                            dw      0               ; hidden sector (high)
                                            dw      0               ; number of sectors (low)
                                            dw      0               ; number of sectors (high)
                                            ; 09/12/2023
                                            ; FAT32 extensions (to BDS)
                                            times   28 db 0
                                            ;
                                            db 90h
                                ;align 2

                                ;align 2
                                ;BPB144:
                                ;bpb144:    ; Retro DOS v4.0 feature only !      ; 1.44MB diskettes
                                ;
                                ;           dw      512             ; physical sector size in bytes
                                ;           db      1               ; sectors/allocation unit
                                ;           dw      1               ; reserved sectors for dos
                                ;           db      2               ; number of allocation tables
                                ;           dw      224             ; number of directory entries
                                ;           dw      2880 ; 2*18*80  ; number of sectors (at 512 bytes each)
                                ;           db      0F0h            ; media descriptor
                                ;           dw      9               ; number of fat sectors
                                ;           dw      18              ; sectors per track
                                ;           dw      2               ; heads
                                ;           dw      0               ; hidden sector count (low word)
```

```
                                ;           dw      0               ; hidden sector (high)
                                ;           dw      0               ; number of sectors (low)
                                ;           dw      0               ; number of sectors (high)
                                ;
                                ;           db 90h
                                ;align 2

                                BPB288:
                                ;bpb288:    ; bpbx <512, 2, 1, 2, 240, 5760, 0F0h, 9, 36, 2, 0, 0, 0, 0>
                                            ; 3.5" diskettes - 2.88 MB ;
                                            dw      512             ; physical sector size in bytes
                                            db      2               ; sectors/allocation unit
                                            dw      1               ; reserved sectors for dos
                                            db      2               ; number of allocation tables
                                            dw      240             ; number of directory entries
                                            dw      5760 ; 2*36*80  ; number of sectors (at 512 bytes each)
                                            db      0F0h            ; media descriptor
                                            dw      3               ; number of fat sectors
                                            dw      9               ; sectors per track
                                            dw      2               ; heads
                                            dw      0               ; hidden sector count (low word)
                                            dw      0               ; hidden sector (high)
                                            dw      0               ; number of sectors (low)
                                            dw      0               ; number of sectors (high)
                                            ; 09/12/2023
                                            ; FAT32 extensions (to BDS)
                                            times   28 db 0
                                            ;
                                            db 90h
                                ;align 2

                                %endif

                                ; ----------------------------------------------------------------------------
                                                                    ; align 2
                                ; 09/12/2022
                                %if 0
                                bpbtable:   dw bpb48t               ; 48tpi drives
                                            dw bpb96t               ; 96tpi drives
                                            dw bpb35                ; 3.5" drives
                                            dw bpb35                ; unused 8" diskette
                                            dw bpb35                ; unused 8" diskette
                                            dw bpb35                ; used for hard       disk
                                            dw bpb35                ; used for tape       drive
                                            dw bpb35                ; FFOTHER
                                            dw bpb35                ; ERIMO
                                            dw bpb288               ; 2.88MB drive
                                            ;
                                            ;dw bpb144              ; 1.44MB drive - Retro DOS v4.0 feature !
                                %endif

                                ; 13/12/2022
                                %if 0
                                BPBTABLE:   dw BPB48T               ; 48tpi drives
                                            dw BPB96T               ; 96tpi drives
                                            dw BPB35                ; 3.5" drives
                                            dw BPB35                ; unused 8" diskette
                                            dw BPB35                ; unused 8" diskette
                                            dw BPB35                ; used for hard       disk
                                            dw BPB35                ; used for tape       drive
                                            dw BPB35                ; FFOTHER
                                            dw BPB35                ; ERIMO
                                            dw BPB288               ; 2.88MB drive
                                            ;
                                            ;dw BPB144              ; 1.44MB drive - Retro DOS v4.0 feature !

                                %endif

                                ; ----------------------------------------------------------------------------

                                ;   entry point to call utility functions in Bios_Code. At this time,
                                ;     we aren't doing any A20 switching. During MSINIT time Bios_Code
                                ;     will not yet be moved to its final resting place, so we know
                                ;     it'll be low.
                                ;
                                ;   to use this function, do a "push cs" and load bp with the offset of
                                ;     the function you want to call in Bios_Code. This routine will
                                ;     push the address of a retf in Bios_Code onto the stack which
                                ;     will get executed when the utility function finishes. It will
                                ;     then transfer control to Bios_Code:bp using a couple of pushes
                                ;     and a retf

                                ; 16/10/2022
                                ;BC_RETF equ bc_retf - DOSBIOSEG_2C7h
                                ; 09/12/2022
                                BC_RETF equ bc_retf

                                ; 09/12/2023
                                ;PCDOS 7.1 IBMBIO.COM bc_retf offset = 0CAh (in BIOSCODE segment = 364h)

                                addr_of_bcretf:    ;dw 0C8h             ; dw bc_retf
                                                                        ; 2C7h:0C8h = 70h:2638h
                                                                        ; 09/12/2023
                                                                        ; 364h:0CAh = 70h:300Ah ; PCDOS 7.1
00001A6F [CA00]                                 dw BC_RETF              ; dw 0CAh

                                ; ----------------------------------------------------------------------------

                                call_bios_code:    ; proc far
00001A71 2EFF36[6F1A]                           push    word [cs:addr_of_bcretf]
                                                                        ; set up near return to far return
00001A76 2EFF36[0406]                           push    word [cs:cdev+2] ; push Bios_Code segment
00001A7B 55                                     push    bp              ; save offset of utility function
00001A7C CB                                     retf                    ; far jump to (DOS)BIOS code

                                ; ----------------------------------------------------------------------------

                                                ; 09/12/2023 - Retro DOS v5.0 IBMBIO.COM/IO.SYS
                                                ; 20/12/2022
00001A7D 00                     flp_drvs:   db 0
                                                ; 11/12/2023
                                                ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:1B81h)
                                firstcluster_hw:
00001A7E 0000                                   dw 0    ; 06/04/2024
00001A80 00                     Boot_Drv:   db 0

                                ; ----------------------------------------------------------------------------

                                ; 09/12/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
                                ;--------------------------------------------------------------------------
                                ; PCDOS 7.1 CD BOOT option code
                                ;--------------------------------------------------------------------------
                                ; (PCDOS 7.1 IBMBIO.COM, BIOSDATA:1B84h)
```

```
                                cd_boot_option:
00001A81 50                             push    ax
00001A82 1E                             push    ds
00001A83 06                             push    es
00001A84 52                             push    dx
                                cdbo_1:
00001A85 B401                           mov     ah, 1
00001A87 CD16                           int     16h                     ; KEYBOARD - status
00001A89 7406                           jz      short cdbo_2
00001A8B 30E4                           xor     ah, ah
00001A8D CD16                           int     16h                     ; KEYBOARD - READ CHAR FROM BUFFER, WAIT IF EMPTY
                                                                        ; Return: AH = scan code, AL = character
00001A8F EBF4                           jmp     short cdbo_1
                                cdbo_2:
00001A91 0E                             push    cs
00001A92 1F                             pop     ds
00001A93 BE[6B1B]                       mov     si, cd_boot_msg               ; "Press the ENTER key to boot from CD"...
00001A96 AC                             lodsb
                                cdbo_3:
00001A97 BB0700                         mov     bx, 7
00001A9A B40E                           mov     ah, 0Eh
00001A9C CD10                           int     10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
                                                                        ; AL = character, BH = display page (alpha modes)
                                                                        ; BL = foreground color (graphics modes)
00001A9E AC                             lodsb
00001A9F 08C0                           or      al, al
00001AA1 75F4                           jnz     short cdbo_3
00001AA3 B84000                         mov     ax, 40h
00001AA6 8ED8                           mov     ds, ax
                                        ;mov    bx, [6Ch]               ; 0:46Ch = Daily timer counter (4 bytes)
                                        ; 09/12/2023
00001AA8 8B166C00                       mov     dx, [6Ch]
00001AAC 8B366E00                       mov     si, [6Eh]
                                wait_for_key:
                                        ;push   bx
                                        ;mov    bx, 7
                                        ; bx = 7
00001AB0 B8080E                         mov     ax, 0E08h
00001AB3 CD10                           int     10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
                                                                        ; AL = character, BH = display page (alpha modes)
                                                                        ; BL = foreground color (graphics modes)
00001AB5 B8200E                         mov     ax, 0E20h
00001AB8 CD10                           int     10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
                                                                        ; AL = character, BH = display page (alpha modes)
                                                                        ; BL = foreground color (graphics modes)
00001ABA B8080E                         mov     ax, 0E08h
00001ABD CD10                           int     10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
                                                                        ; AL = character, BH = display page (alpha modes)
                                                                        ; BL = foreground color (graphics modes)
                                        ;pop    bx
                                        ;add    bx, 18                  ; 18.2 ticks per second
                                        ; 09/12/2023
00001ABF 83C212                         add     dx, 18
00001AC2 83D600                         adc     si, 0                   ; next second (if carry flag is 1)
                                continue_to_wait:
00001AC5 B401                           mov     ah, 1
00001AC7 CD16                           int     16h                     ; KEYBOARD - status
00001AC9 741B                           jz      short cdbo_5
00001ACB B400                           mov     ah, 0
00001ACD CD16                           int     16h                     ; KEYBOARD - READ CHAR FROM BUFFER, WAIT IF EMPTY
                                                                        ; Return: AH = scan code, AL = character

                                        ; 09/12/2023
                                        ;cmp    ax, 11Bh ; ESC key
                                        ;jz     short cdb0_7
                                ;cdbo_4:
                                        ;push   ax ; *
00001ACF 89C2                           mov     dx, ax ; *

                                        ; CRLF (next line)
                                        ;mov    bx, 7
                                        ; bx = 7
00001AD1 B80D0E                         mov     ax, 0E0Dh
00001AD4 CD10                           int     10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
                                                                        ; AL = character, BH = display page (alpha modes)
                                                                        ; BL = foreground color (graphics modes)
00001AD6 B80A0E                         mov     ax, 0E0Ah
00001AD9 CD10                           int     10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
                                                                        ; AL = character, BH = display page (alpha modes)
                                                                        ; BL = foreground color (graphics modes)
                                        ; 09/12/2023
                                        ;pop    ax ; *

00001ADB 81FA1B01                       cmp     dx, 11Bh
                                        ;cmp    ax, 11Bh ; ESC key (to cancel CD/DVD boot)
00001ADF 7418                           je      short cdbo_7

                                cdbo_4:         ; 10/12/2023
00001AE1 5A                             pop     dx
00001AE2 07                             pop     es
00001AE3 1F                             pop     ds
00001AE4 58                             pop     ax
00001AE5 C3                             retn
                                cdbo_5:
00001AE6 3B366E00                       cmp     si, [6Eh]
00001AEA 7504                           jnz     short cdbo_6
                                        ; 09/12/2023
00001AEC 3B166C00                       cmp     dx, [6Ch]
                                        ;cmp    bx, [6Ch]
                                cdbo_6:
00001AF0 73D3                           jnb     short continue_to_wait
00001AF2 2EFE0E[6A1B]                   dec     byte [cs:time_counter]
00001AF7 75B7                           jnz     short wait_for_key
                                cdbo_7:
                                        ; 09/12/2023
                                        ; CRLF (next line)
                                        ;
                                        ;mov    bx, 7
                                        ;mov    ax, 0E0Dh
                                        ;int    10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
                                        ;                               ; AL = character, BH = display page (alpha modes)
                                        ;                               ; BL = foreground color (graphics modes)
                                        ;mov    ax, 0E0Ah
                                        ;int    10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
                                        ;                               ; AL = character, BH = display page (alpha modes)
                                        ;                               ; BL = foreground color (graphics modes)

00001AF9 0E                             push    cs
00001AFA 1F                             pop     ds
                                        ; 09/12/2023
00001AFB 1E                             push    ds
00001AFC 07                             pop     es
                                        ; es = ds = cs
```

```
6083 00001AFD B8004B                    mov     ax, 4B00h
6084                                    ;xor    dl, dl
6085                                    ; 09/12/2023
6086 00001B00 31D2                      xor     dx, dx
6087                                    ; dl = disk drive = 0  ; fd
6088                                    ;mov    si, 1C93h
6089 00001B02 BE[571B]                  mov     si, empty_dap_buff
6090 00001B05 CD13                      int     13h                     ; DISK - Bootable CD-ROM - AL = TERMINATE DISK EMULATION
6091                                    ; DS:SI = Specification packet filled
6092
6093                                    ;mov    dx, 80h
6094                                    ;xor    ax, ax
6095                                    ; 09/12/2023
6096 00001B07 B81300                    mov     ax, 19
6097 00001B0A 89F7                      mov     di, si
6098                                    ;mov    byte [si], 13h
6099                                    ;mov    [si+1], al
6100 00001B0C AB                        stosw
6101                                    ;mov    [si+2], dx
6102 00001B0D B080                      mov     al, 80h
6103 00001B0F AB                        stosw
6104 00001B10 89C2                      mov     dx, ax
6105                                    ;mov    [si+4], ax
6106                                    ;mov    [si+6], ax
6107                                    ;mov    [si+8], ax
6108                                    ;mov    [si+0Ah], ax
6109                                    ;mov    [si+0Ch], ax
6110                                    ;mov    [si+0Eh], ax
6111                                    ;mov    [si+10h], al
6112                                    ;mov    [si+11h], al
6113                                    ;mov    [si+12h], al
6114 00001B12 B90F00                    mov     cx, 15
6115 00001B15 F3AA                      rep     stosb
6116                                    ; dl = disk drive = 80h ; hd
6117 00001B17 B8004B                    mov     ax, 4B00h
6118 00001B1A CD13                      int     13h                     ; DISK - Bootable CD-ROM - AL = TERMINATE DISK EMULATION
6119 00001B1C 31C0                      xor     ax, ax
6120                                    ; 09/12/2023
6121                                    ;mov    dx, 80h
6122                                    ; dx = 80h
6123 00001B1E CD13                      int     13h                     ; DISK - RESET DISK SYSTEM
6124                                                                    ; DL = drive (if bit 7 is set both hard disks and floppy disks reset)
6125                                    ; 09/12/2023
6126                                    ;push   cs
6127                                    ;pop    es
6128                                    ; es = ds = cs
6129
6130 00001B20 B80102                    mov     ax, 201h
6131                                    ;mov    bx, 152h
6132 00001B23 BB[5201]                  mov     bx, disksector
6133                                    ;mov    cx, 1
6134                                    ; 09/12/2023
6135 00001B26 41                        inc     cx ; cx = 1
6136                                    ;mov    dx, 80h
6137                                    ; dx = 80h
6138 00001B27 CD13                      int     13h                     ; DISK - READ SECTORS INTO MEMORY
6139                                                                    ; AL = number of sectors to read, CH = track, CL = sector
6140                                                                    ; DH = head, DL = drive, ES:BX -> buffer to fill
6141                                                                    ; Return: CF set on error, AH = status, AL = number of sectors read
6142                                    ;jc     short cdbo_8
6143                                    ; 10/12/2023
6144 00001B29 72B6                      jc      short cdbo_4
6145
6146 00001B2B 2681BFFE0155AA            cmp     word [es:bx+1FEh], 0AA55h
6147                                    ;jz     short cdbo_9
6148                                    ; 10/12/2023
6149 00001B32 75AD                      jnz     short cdbo_4
6150                                ;cdbo_8:
6151                                    ;jmp    short cdbo_4
6152                                ;cdbo_9:
6153                                    ; 10/12/2023
6154                                    ; (stack clearing -pop- is not necessary here,
6155                                    ;  PCDOS 7.1 boot sector will set stack pointer again)
6156                                    ;pop    ax ; near call return address
6157                                    ;pop    cx ; +++ ; ch = [MediaByte]
6158
6159                                    ; 09/12/2023
6160                                    ;push   cs
6161                                    ;pop    ds
6162                                    ; ds = cs
6163 00001B34 31C0                      xor     ax, ax  ; 0
6164 00001B36 BF007C                    mov     di, 7C00h
6165 00001B39 8EC0                      mov     es, ax
6166 00001B3B 89DE                      mov     si, bx
6167 00001B3D 06                        push    es
6168 00001B3E 57                        push    di
6169 00001B3F B90001                    mov     cx, 100h ; 256
6170                                    ; 10/12/2023
6171                                    ;cld    ; not necessary (direction flag is already cleared)
6172 00001B42 F3A5                      rep movsw
6173 00001B44 8ED8                      mov     ds, ax
6174 00001B46 BE7800                    mov     si, 78h
6175 00001B49 2EA1[121A]                mov     ax, [cs:Orig_Int1Eh_Table]
6176 00001B4D 8904                      mov     [si], ax
6177 00001B4F 2EA1[141A]                mov     ax, [cs:Orig_Int1Eh_Table+2]
6178 00001B53 894402                    mov     [si+2], ax
6179 00001B56 CB                        retf
6180
6181                                ; ---------------------------------------------------------------------------
6182                                dap_buffer: ; 16/12/2023
6183
6184 00001B57 13                    empty_dap_buff:     db 19
6185                                                    ;db 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0  ; db 18 dup(0)
6186 00001B58 00<rep 12h>                               times 18 db 0
6187 00001B6A 05                    time_counter:       db 5    ; 5 seconds
6188 00001B6B 0D0A                  cd_boot_msg:        db 0Dh,0Ah
6189                                                    ;db 'Press the ENTER key to boot from CD or DVD......',0
6190                                                    ; 09/12/2023
6191 00001B6D 507265737320616E79-                       db 'Press any key to boot from CD or DVD ...',0
6191 00001B76 206B657920746F2062-
6191 00001B7F 6F6F742066726F6D20-
6191 00001B88 4344206F7220445644-
6191 00001B91 202E2E2E00
6192
6193                                ; ---------------------------------------------------------------------------
6194
6195                                ; 01/10/2022 - Retro DOS v4.0 (MSDOS 5.0)
6196
6197                                ; 10/12/2023 - Retro DOS v5.0 (PCDOS 7.1)
6198                                ; (PCDOS 7.1 IBMBIO.COM, BIOSDATA:1CDAh)
6199
6200                                ;---------------------------------------------------------------------------
6201                                ; entry point from boot sector
6202                                ;---------------------------------------------------------------------------
```

```
                                ; 27/12/2018
                        init:
                                ; MSDOS 6.0 (MSINIT.ASM)
                                ;============================================================
                                ;
                                ; entry from boot sector. the register contents are:
                                ;
                                ;   dl = int 13 drive number we booted from
                                ;   ch = media byte
                                ;   bx = first data sector on disk.
                                ;   ax = first data sector (high)
                                ;   di = sectors/fat for the boot media.

                                ; 07/04/2018
                                ;============================================================
                                ; Retro DOS v2.0 - registers from FD Boot Sector
                                   ; DL = [bsDriveNumber]
                                ; DH = [bsMedia]
                                ; AX = [bsSectors] ; Total sectors
                                ; DS = 0, SS = 0
                                ; BP = 7C00h

                                ; 10/12/2023
                                ; Retro DOS v5.0 (IBMBIO.COM)
                                ;============================================================
                                ; PCDOS 7.1 IBMBIO.COM - registers from MSLOAD section
                                   ; DL = [BootDrive]
                                ; CH = [MediaByte]
                                ; AX:BX = First data Sector
                                ; DS:SI = Original INT 1Eh table address
                                ;
                                ; Stack: INT 1Eh vector (0:78h) !not used! (dword [sp])
                                ;        INT 1Eh table address !not used! (dword [sp+4])
                                ; DI = 78h !not used!

                                ; 11/12/2023
                                ;cli    ; not necessary at this stage

                                        ; 10/12/2023
                                        ;mov    [cs:Orig_Int1Eh_Table+2], ds
                                        ;mov    [cs:Orig_Int1Eh_Table], si
 00001B96 1E                            push    ds
 00001B97 07                            pop     es
 00001B98 0E                            push    cs
 00001B99 1F                            pop     ds
 00001B9A 8C06[141A]                    mov     [Orig_Int1Eh_Table+2], es
 00001B9E 8936[121A]                    mov     [Orig_Int1Eh_Table], si

                                        ; 21/12/2022
                                        ; ds = 0 (?)
                                        ;push   ax
                                        ;xor    ax, ax
                                        ;mov    ds, ax
                                        ;pop    ax

                                ; 02/10/2022
                                ; ------------------------------------------------------------------------------
                                ; Note: Retro DOS v4.0 Kernel does not use/contain MSLOAD part of IO.SYS (5.0)
                                ;   Because, Retro DOS v2 boot sector loads complete/entire MSDOS.SYS
                                ;   (RETRODOS.SYS) Kernel file (IO.SYS & MSDOS.SYS together).
                                ;   As result of boot sector ve init differences, Retro DOS init code (here)
                                ;   moves kernel to segment 070h at first, then sets diskette parameters
                                ;   at segment 50h (while MSDOS 5.0 boot sector sets this).
                                ; ------------------------------------------------------------------------------

                                ; msload will check the extended boot record and set ax, bx accordingly.

                                ;   msload passes a 32 bit sector number hi word in ax and low in bx
                                ;   save this in cs:bios_h and cs:bios_l. this is for the start of
                                ;   data sector of the bios.

                                        ;mov    [cs:bios_h], ax       ; (start of) dos bios (IO.SYS) data sector
                                        ;mov    [cs:bios_l], bx
                                        ; 10/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                        ;mov    [cs:First_Data_Sector+2], ax
                                        ;mov    [cs:First_Data_Sector], bx
                                        ;mov    [cs:Boot_Drv], dl
                                        ; ds = cs
 00001BA2 A3[FE19]                      mov     [First_Data_Sector+2], ax
 00001BA5 891E[FC19]                    mov     [First_Data_Sector], bx
 00001BA9 8816[801A]                    mov     [Boot_Drv], dl

                                ; with the following information from msload, we don't need the
                                ;     boot sector any more.-> this will solve the problem of 29 kb size
                                ;     limitation of msbio.com file.

 00001BAD 0E                            push    cs              ; Save a peck of interrupt vectors...
 00001BAE 07                            pop     es

 00001BAF 51                            push    cx ; +++ ; ch = [MediaByte]
                                        ;push   di ; *! (not necessary) ; 10/12/2023

                                        ; 10/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
 00001BB0 FC                            cld     ; (may not be necessary)

 00001BB1 31C0                          xor     ax, ax
 00001BB3 8ED8                          mov     ds, ax  ; ds = 0

                                        ; 06/04/2024
 00001BB5 50                            push    ax ; push ds ; 0

                                        ;mov    ax, 544h        ; SYSINIT segment
 00001BB6 B80405                        mov     ax, SYSINITSEG

                                        ; (PCDOS 7.1 IBMBIO.COM, BIOSDATA:1D06h)

                                        ; check (1st sector) of the root directory -of BOOT CD-
                                        ; for special names (as boot option signature)

 00001BB9 BE4005                        mov     si, 540h  ; ROOT DIRECTORY BUFFER offset 40h
                                                          ; (BOOT DRV's root directory the 3rd entry)
                                chk_boot_hdnoz:
 00001BBC 803C00                        cmp     byte [si], 0
 00001BBF 7436                          jz      short chk_no_logo_noz
 00001BC1 813C5F42                      cmp     word [si], 425Fh ; '_BOOT_HDNOZ'
 00001BC5 7527                          jnz     short chk_next_1
 00001BC7 817C024F4F                    cmp     word [si+2], 4F4Fh ; 'OO'
 00001BCC 7520                          jnz     short chk_next_1
 00001BCE 817C04545F                    cmp     word [si+4], 5F54h
 00001BD3 7519                          jnz     short chk_next_1
 00001BD5 817C064844                    cmp     word [si+6], 4448h ; 'HD'
 00001BDA 7512                          jnz     short chk_next_1
 00001BDC 817C084E4F                    cmp     word [si+8], 4F4Eh
 00001BE1 750B                          jnz     short chk_next_1
```

```
6327 00001BE3 807C0A5A                  cmp     byte [si+0Ah], 5Ah ; 'Z'
6328 00001BE7 7505                      jnz     short chk_next_1
6329 00001BE9 E895FE                    call    cd_boot_option
6330 00001BEC EB09                      jmp     short chk_no_logo_noz
6331                                chk_next_1:
6332 00001BEE 83C620                    add     si, 32  ; (next entry)
6333 00001BF1 81FE0007                  cmp     si, 700h
6334 00001BF5 72C5                      jb      short chk_boot_hdnoz
6335                                chk_no_logo_noz:
6336 00001BF7 BE4005                    mov     si, 540h  ; (BOOT DRV's root directory the 3rd entry)
6337                                chk_no_logo_noz2_nxt:
6338 00001BFA 803C00                    cmp     byte [si], 0
6339 00001BFD 7431                      jz      short write_start_msg
6340 00001BFF 813C4E4F                  cmp     word [si], 4F4Eh ; 'NO_LOGO NOZ'
6341 00001C03 7522                      jnz     short chk_next_2
6342 00001C05 817C025F4C                cmp     word [si+2], 4C5Fh
6343 00001C0A 751B                      jnz     short chk_next_2
6344 00001C0C 817C044F47                cmp     word [si+4], 474Fh
6345 00001C11 7514                      jnz     short chk_next_2
6346 00001C13 817C064F20                cmp     word [si+6], 204Fh
6347 00001C18 750D                      jnz     short chk_next_2
6348 00001C1A 817C084E4F                cmp     word [si+8], 4F4Eh
6349 00001C1F 7506                      jnz     short chk_next_2
6350 00001C21 807C0A5A                  cmp     byte [si+0Ah], 5Ah
6351 00001C25 741C                      jz      short startmsg_ok
6352                                chk_next_2:
6353 00001C27 83C620                    add     si, 32  ; (next entry)
6354 00001C2A 81FE0007                  cmp     si, 700h
6355 00001C2E 72CA                      jb      short chk_no_logo_noz2_nxt
6356                                write_start_msg:
6357 00001C30 8ED8                      mov     ds, ax  ; SYSINIT segment
6358 00001C32 BE[C751]                  mov     si, StartMsg   ; "Starting PC DOS...\r\n\n"
6359                                startmsg_nxt_chr:
6360 00001C35 AC                        lodsb
6361 00001C36 08C0                      or      al, al
6362 00001C38 7409                      jz      short startmsg_ok
6363 00001C3A B40E                      mov     ah, 0Eh
6364 00001C3C BB0700                    mov     bx, 7
6365 00001C3F CD10                      int     10h             ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
6366                                                            ; AL = character, BH = display page (alpha modes)
6367                                                            ; BL = foreground color (graphics modes)
6368 00001C41 EBF2                      jmp     short startmsg_nxt_chr
6369                                startmsg_ok:
6370                                    ; 06/04/2024
6371 00001C43 1F                        pop     ds ; 0
6372
6373                                    ; 10/12/2023
6374                                    ; ds = 0
6375
6376                                    ; 21/12/2022
6377                                    ; ds = 0 (?)
6378                                    ; 24/12/2022
6379                                    ; ds = cs
6380                                    ;xor    cx, cx
6381                                    ;mov    ds, cx
6382                                    ; ds = 0
6383
6384                                    ;mov    cl, 5
6385                                    ; 10/12/2023
6386 00001C44 B90500                    mov     cx, 5           ; NUMROMVECTORS
6387                                                            ; no. of rom vectors to      be saved
6388                                    ;mov    si, offset RomVectors ; point to list of int vectors
6389 00001C47 BE[0001]                  mov     si, RomVectors
6390
6391                                    ; 10/12/2023
6392 00001C4A FA                        cli
6393                                next_int_:
6394 00001C4B 2E                        cs      ; 16/10/2022
6395 00001C4C AC                        lodsb
6396                                    ;lods   byte ptr cs:[si] ; cs lodsb
6397 00001C4D 98                        cbw                     ; ax = interrupt number
6398 00001C4E D1E0                      shl     ax, 1
6399 00001C50 D1E0                      shl     ax, 1           ; int no * 4
6400 00001C52 89C7                      mov     di, ax          ; interrupt vector address
6401 00001C54 87FE                      xchg    si, di          ; rombios interrupt vector address in si
6402                                                            ; saving address in di
6403                                    ;lodsw                  ; movsw
6404                                    ;stosw
6405                                    ;lodsw                  ; movsw
6406                                    ;stosw                  ; save the vector
6407                                    ; 21/04/2024
6408 00001C56 A5                        movsw
6409 00001C57 A5                        movsw
6410
6411 00001C58 87FE                      xchg    si, di
6412 00001C5A E2EF                      loop    next_int_
6413
6414                                    ; 10/12/2023
6415                                    ;pop    di  ; *!
6416                                    ;pop    cx ; +++ ; ch = [MediaByte]
6417
6418                                ; we need to save int13 in two places in case we are running on an at.
6419                                ; on ats we install the ibm supplied rom_bios patch which hooks
6420                                ; int13 ahead of orig13. since int19 must unhook int13 to point to the
6421                                ; rom int13 routine, we must have that rom address also stored away.
6422
6423                                    ; 21/12/2022
6424                                    ;mov    ax, [cs:Old13] ; save old13 in orig13 also
6425                                    ;mov    [cs:Orig13], ax
6426                                    ;mov    ax, [cs:Old13+2]
6427                                    ;mov    [cs:Orig13+2], ax
6428
6429                                    ; 16/10/2022
6430 00001C5C C7064C00[ED06]              mov     word [13h*4], block13
6431                                    ;mov    word ptr ds:4Ch, offset      block13 ; 13h*4
6432                                                            ; set up int 13       for new action
6433 00001C62 8C0E4E00                  mov     [13h*4+2], cs
6434                                    ;mov    word ptr ds:4Eh, cs ; 13h*4+2
6435 00001C66 C7065400[9907]            mov     word [15h*4], Int15
6436                                    ;mov    word ptr ds:54h, offset      Int15 ; 15h*4
6437                                                            ; set up int 15       for new action
6438 00001C6C 8C0E5600                  mov     [15h*4+2], cs
6439                                    ;mov    word ptr ds:56h, cs ; 15h*4+2
6440 00001C70 C7066400[5907]            mov     word [19h*4], int19
6441                                    ;mov    word ptr ds:64h, offset      int19 ; 19h*4
6442                                                            ; set up int 19       for new action
6443 00001C76 8C0E6600                  mov     [19h*4+2], cs
6444                                    ;mov    word ptr ds:66h, cs ; 19h*4+2
6445
6446                                    ; 10/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
6447 00001C7A A16800                    mov     ax, [68h]       ; 1Ah*4
6448 00001C7D 8B3E6A00                  mov     di, [6Ah]       ; 1Ah*4+2
6449 00001C81 C7066800[AF06]            mov     word [68h], Int1A
6450 00001C87 8C0E6A00                  mov     [6Ah], cs
```

```
6451
6452                                    ; 21/12/2022
6453 00001C8B 0E                       push    cs
6454 00001C8C 1F                       pop     ds
6455
6456                                    ; 10/12/2023
6457 00001C8D A3[AB06]                 mov     [Orig1A], ax
6458 00001C90 893E[AD06]               mov     [Orig1A+2], di
6459
6460 00001C94 A1[0601]                 mov     ax, [Old13]     ; save old13 in orig13 also
6461 00001C97 A3[B400]                 mov     [Orig13], ax
6462 00001C9A A1[0801]                 mov     ax, [Old13+2]
6463 00001C9D A3[B600]                 mov     [Orig13+2], ax
6464                                                    ;
6465 00001CA0 FB                       sti
6466 00001CA1 CD11                     int     11h             ; EQUIPMENT DETERMINATION
6467                                                    ; Return: AX = equipment flag bits
6468                                    ; 10/12/2023
6469                                    ;jmp    short chk_fd_count
6470                                    ; (PCDOS 7.1 IBMBIO.COM, BIOSDATA:1DF7h) ; *!!*
6471                                    ; ((signature))
6472                                    ;push   dx              ; 52h ; 'R'
6473                                    ;push   ax              ; 50h ; 'P'
6474                                    ;push   bx              ; 53h ; 'S'
6475
6476                                ; we have to support a system that does not have any diskette
6477                                ; drives but only hardfiles. this system will ipl from the hardfile.
6478                                ; if the equipment flag bit 0 is 1, then the system has diskette drive(s).
6479                                ; otherwise, the system has only hardfiles.
6480                                ;
6481                                ; important thing is that still, for compatibility reason, the drive letter
6482                                ; for the hardfiles start from "c". so, we still need to allocate dummy bds
6483                                ; drive a and drive b. at sysinit time, we are going to set cds table entry
6484                                ; of dpb pointer for these drives to 0, so any user attempt to access this
6485                                ; drives will get "invalid drive letter ..." message. we are going to
6486                                ; establish "fakefloppydrv" flag. ***sysinit module should call int 11h to
6487                                ; determine whether there are any diskette drivers in the system or not.!!!***
6488
6489                                ; check the register returned by the equipment determination interrupt
6490                                ; we have to handle the case of no diskettes in the system by faking
6491                                ; two dummy drives.
6492                                ;
6493                                ; if the register indicates that we do have floppy drives we don't need
6494                                ; to do anything special.
6495                                ;
6496                                ; if the register indicates that we don't have any floppy drives then
6497                                ; what we need to do is set the fakefloppydrv variable, change the
6498                                ; register to say that we do have floppy drives and then go to execute
6499                                ; the code which starts at notsingle. this is because we can skip the
6500                                ; code given below which tries to find if there are one or two drives
6501                                ; since we already know about this.
6502
6503                                chk_fd_count:   ; 10/12/2023
6504                                        ;or     ax, 1   ; *!!*
6505
6506                                        ; 12/12/2022
6507 00001CA3 A801                       test    al, 1
6508                                     ;test   ax, 1           ; floppy drives       present?
6509 00001CA5 7525                     jnz     short normalfloppydrv ;       yes.
6510
6511                                ; Some ROM BIOSs lie that there are no floppy drives. Lets find out
6512                                ; whether it is an old ROM BIOS or a new one
6513                                ;
6514                                ; WARNING !!!
6515                                ;
6516                                ; This sequence of code is present in SYSINIT1.ASM also. Any modification
6517                                ; here will require an equivalent modification in SYSINIT1.ASM also
6518
6519                                        ; 10/12/2023
6520                                        ; ((cx is already on top of the stack))
6521                                        ;push   cx      ; +++ ; ch = [MediaByte]
6522                                        ;push   bx      ; not necessary
6523 00001CA7 50                       push    ax
6524 00001CA8 52                       push    dx
6525                                    ;push   di      ; not necessary
6526 00001CA9 06                       push    es
6527
6528 00001CAA B408                     mov     ah, 8
6529 00001CAC B200                     mov     dl, 0
6530 00001CAE CD13                     int     13h             ; DISK - DISK -       GET CURRENT DRIVE PARAMETERS (XT,AT,XT286,CONV,PS)
6531                                                    ; DL = drive number
6532                                                    ; Return: CF set on error, AH =      status code, BL       = drive type
6533                                                    ; DL = number of consecutive drives
6534                                                    ; DH = maximum value for head number, ES:DI -> drive parameter
6535 00001CB0 7204                     jc      short _gdskp_error
6536                                    ;mov    [cs:flp_drvs], dl
6537                                    ; 21/12/2022
6538                                    ; ds = cs
6539 00001CB2 8816[7D1A]               mov     [flp_drvs], dl
6540                                _gdskp_error:
6541                                    ; 10/12/2023
6542 00001CB6 07                       pop     es
6543                                    ;pop    di
6544 00001CB7 5A                       pop     dx
6545 00001CB8 58                       pop     ax
6546                                    ;pop    bx
6547 00001CB9 59                       pop     cx      ; +++ ; ch = [MediaByte]
6548 00001CBA 7210                     jc      short normalfloppydrv
6549                                                    ; if error it is an old ROM BIOS
6550                                                    ; so, lets assume that ROM BIOS lied
6551                                    ; 21/12/2022
6552                                    ;cmp    byte [cs:flp_drvs], 0 ; number of drvs == 0?
6553                                    ;jz     short _set_fake_flpdrv
6554                                    ;mov    al, [cs:flp_drvs]
6555 00001CBC A0[7D1A]                 mov     al, [flp_drvs]
6556 00001CBF 08C0                     or      al, al          ; number of drvs == 0?
6557 00001CC1 7403                     jz      short _set_fake_flpdrv
6558
6559                                    ;dec    al              ; make it zero based
6560                                    ; 18/12/2022
6561 00001CC3 48                       dec     ax
6562 00001CC4 EB0A                     jmp     short got_num_flp_drvs
6563                                ; ---------------------------------------------------------------------------
6564
6565                                _set_fake_flpdrv:
6566                                    ; 21/12/2022
6567                                    ;mov    ax, 1
6568                                    ; 10/12/2023
6569 00001CC6 40                       inc     ax ; al = 1
6570 00001CC7 A2[111A]                 mov     [fakefloppydrv], al ; 1
6571                                    ;mov    byte [cs:fakefloppydrv], 1
6572                                    ;                       ; we don't have any floppy drives.
6573                                    ;mov    ax, 1
6574 00001CCA EB09                     jmp     short settwodrive ; well then set it for two drives!
```

```
6575                                    ; ---------------------------------------------------------------------------
6576
6577                                    normalfloppydrv:                ; yes, bit 0 is 1.
6578 00001CCC D0C0                                 rol     al, 1          ; there exist floppy drives.
6579 00001CCE D0C0                                 rol     al, 1          ; put bits 6 & 7 into bits 0 & 1
6580                                    got_num_flp_drvs:
6581                                               ;and    ax, 3          ; only look at bits 0 &      1
6582                                               ; 18/12/2022
6583 00001CD0 2403                                 and     al, 3
6584 00001CD2 7505                                 jnz     short notsingle       ; zero means single drive system
6585 00001CD4 40                                   inc     ax             ; pretend it's a two drive system
6586                                    settwodrive:                            ; set this to two fakedrives
6587                                               ; 21/12/2022
6588                                               ; ds = cs
6589 00001CD5 FE06[7800]                           inc     byte [single]
6590                                               ;inc    byte [cs:single] ; remember this
6591                                    notsingle:
6592 00001CD9 40                                   inc     ax             ; ax has number       of drives, 2-4
6593                                                                       ; is also 0 indexed boot drive if we
6594                                                                       ; booted off hard file
6595 00001CDA 88C1                                 mov     cl, al         ; ch is fat id, cl # floppies
6596
6597                                    ; 16/10/2022
6598                                    ; MSDOS 3.3 - "MSEQU.INC" (24/07/1987)
6599                                    INITSPOT EQU       534h    ; IBM wants 4 zeros here
6600                                    BRKADR      EQU    1BH * 4 ; 6CH, 1BH break vector address
6601                                    TIMADR      EQU    1CH * 4 ; 70H, 1CH timer interrupt
6602                                    DSKADR      EQU    1EH * 4 ; address of ptr to disk parameters
6603                                    SEC9EQU     522h   ; address of disk parameters
6604                                    CHROUT      EQU    29h
6605                                    LSTDRV      EQU     504h
6606
6607                                    ; determine whether we booted from floppy or hard disk...
6608
6609 00001CDC F6C280                               test    dl, 80h        ; boot from floppy ?
6610 00001CDF 7505                                 jnz     short gothrd   ; no.
6611 00001CE1 31C0                                 xor     ax, ax         ; indicate boot       from drive a
6612                                               ; 10/12/2023
6613 00001CE3 A2[801A]                             mov     [Boot_Drv], al
6614                                    gothrd:
6615 00001CE6 31D2                                 xor     dx, dx ; 0     ; ax = 0-based drive we       booted from
6616                                                                       ; bios_l, bios_h set.
6617                                                                       ; cl = number of floppies including fake one
6618                                                                       ; ch = media byte
6619 00001CE8 FA                                   cli
6620 00001CE9 8ED2                                 mov     ss, dx         ; set stack segment and stack pointer
6621 00001CEB BC0007                               mov     sp, 700h
6622 00001CEE FB                                   sti
6623
6624 00001CEF 51                                   push    cx ; *         ; save number of floppies and media byte
6625 00001CF0 88EC                                 mov     ah, ch         ; FAT ID to AH
6626 00001CF2 50                                   push    ax ; **        ; save boot drive number and media byte
6627
6628                                    ; let model_byte, secondary_model_byte be set here!!!
6629
6630 00001CF3 B4C0                                 mov     ah, 0C0h
6631 00001CF5 CD15                                 int     15h            ; SYSTEM - GET CONFIGURATION (XT after 1/10/86,AT mdl 3x9,CONV,XT286,PS)
6632 00001CF7 7215                                 jb      short no_rom_system_conf ; just      use Model_Byte
6633 00001CF9 80FC00                               cmp     ah, 0
6634 00001CFC 7510                                 jnz     short no_rom_system_conf
6635
6636                                               ; 21/12/2022
6637                                               ; ds = cs
6638 00001CFE 268A4702                             mov     al, [es:bx+2]  ; [es:bx+ROMBIOS_DESC.bios_sd_modelbyte]
6639 00001D02 A2[AF05]                             mov     [model_byte], al
6640                                               ;mov    [cs:model_byte], al
6641                                                                       ; get/save model byte
6642 00001D05 268A4703                             mov     al, [es:bx+3]  ; [es:bx+ROMBIOS_DESC.bios_sd_scnd_modelbyte]
6643 00001D09 A2[B005]                             mov     [secondary_model_byte], al
6644                                               ;mov    [cs:secondary_model_byte], al
6645                                                                       ; get/save secondary model byte
6646 00001D0C EB0C                                 jmp     short turn_timer_on
6647                                    ;---------------------------------------------------------------------------
6648
6649                                    no_rom_system_conf:
6650 00001D0E BEFFFF                               mov     si, 0FFFFh
6651 00001D11 8EC6                                 mov     es, si
6652                                               ; 21/12/2022
6653 00001D13 26A00E00                             mov     al, [es:0Eh]   ; get model byte (from 0FFFFh:0Eh)
6654 00001D17 A2[AF05]                             mov     [model_byte], al
6655                                               ;mov    [cs:model_byte], al ; save model byte
6656                                    turn_timer_on:
6657 00001D1A B020                                 mov     al, 20h ; ' '   ; turn on the timer
6658 00001D1C E620                                 out     20h, al        ; Interrupt controller,       8259A.
6659                                                                       ; AKPORT
6660
6661                                    ; some olivetti m24 machines have an 8530 serial communications
6662                                    ; chip installed at io address 50h and 52h. if we're running
6663                                    ; on one of those, we must inhibit the normal aux port initialization
6664
6665                                               ; 21/12/2022
6666                                               ; ds = cs
6667 00001D1E 803E[AF05]00                          cmp     byte [model_byte], 0
6668                                               ;cmp    byte [cs:model_byte], 0 ; next to last       byte in rom bios
6669 00001D23 7510                                 jnz     short not_olivetti_m24 ; skip for all other machines
6670                                                                       ; (except olivetti m24)
6671 00001D25 E466                                 in      al, 66h        ; is 8530 installed?
6672 00001D27 A820                                 test    al, 20h
6673 00001D29 740A                                 jz      short not_olivetti_m24 ; we're done if not
6674 00001D2B B00F                                 mov     al, 0Fh        ; double check
6675 00001D2D E650                                 out     50h, al
6676 00001D2F E450                                 in      al, 50h
6677 00001D31 A801                                 test    al, 1          ; this test was       copied from olivetti
6678 00001D33 7414                                 jz      short skip_aux_port_init ; take      this branch if 8530 installed
6679
6680                                    not_olivetti_m24:
6681 00001D35 B003                                 mov     al, 3          ; init com4
6682 00001D37 E8F00A                               call    aux_init
6683 00001D3A B002                                 mov     al, 2          ; init com3
6684 00001D3C E8EB0A                               call    aux_init
6685 00001D3F B001                                 mov     al, 1          ; init com2
6686 00001D41 E8E60A                               call    aux_init
6687 00001D44 30C0                                 xor     al, al         ; init com1
6688 00001D46 E8E10A                               call    aux_init
6689
6690                                    skip_aux_port_init:
6691 00001D49 B002                                 mov     al, 2          ; init lpt3
6692 00001D4B E8D40A                               call    print_init
6693 00001D4E B001                                 mov     al, 1          ; init lpt2
6694 00001D50 E8CF0A                               call    print_init
6695 00001D53 30C0                                 xor     al, al         ; init lpt1
6696 00001D55 E8CA0A                               call    print_init
6697
6698 00001D58 31D2                                 xor     dx, dx ; 0
```

```
00001D5A 8EDA                           mov     ds, dx          ; to initialize       print screen vector
00001D5C 8EC2                           mov     es, dx

                                        ; 11/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
00001D5E BF3405                         mov     di, 534h        ; offset INITSPOT
                                        ;mov    di, INITSPOT    ; 0534h
                                                                ; IBMDOS.COM's first cluster - high word
                                                                ; 520h (the 2nd entry of root dir) + 14h
00001D61 8B05                           mov     ax, [di]
                                        ;mov    [firstcluster_hw], ax
                                        ; 06/04/2024
00001D63 2EA3[7E1A]                     mov     [cs:firstcluster_hw], ax

00001D67 31C0                           xor     ax, ax
                                        ; 11/12/2023
                                        ; 16/10/2022
                                        ;mov    di, INITSPOT    ; 0534h
                                        ;;mov   di, 534h        ; INITSPOT (0000h:0534h)
                                                                ; IBM wants 4 zeros here
00001D69 AB                             stosw
00001D6A AB                             stosw
00001D6B 8CC8                           mov     ax, cs          ; fetch segment
00001D6D C7066C00[0E06]                 mov     word [BRKADR], cbreak
                                        ;mov    word ptr ds:6Ch, offset       cbreak ; [BRKADR]
                                                                ; break entry point
00001D73 A36E00                         mov     [BRKADR+2], ax
                                        ;mov    ds:6Eh, ax      ; vector for break
00001D76 C706A400[8206]                 mov     word [CHROUT*4], outchr
                                        ;mov    word ptr ds:0A4h, offset outchr       ; [CHROUT*4]
00001D7C A3A600                         mov     [CHROUT*4+2], ax
                                        ;mov    ds:0A6h, ax     ; [CHROUT*4+2]
00001D7F BF0400                         mov     di, 4
00001D82 BB[1406]                       mov     bx, intret ; 19/10/2022
                                        ;mov    bx, offset intret ; intret (cs:intret)
                                                                ; will initialize rest of interrupts
00001D85 93                             xchg    ax, bx
00001D86 AB                             stosw                   ; location 4
00001D87 93                             xchg    ax, bx          ; cs:
00001D88 AB                             stosw                   ; int 1 ; location 6
00001D89 83C704                         add     di, 4           ; skip int 2
00001D8C 93                             xchg    ax, bx
00001D8D AB                             stosw                   ; location 12
00001D8E 93                             xchg    ax, bx          ; cs:
00001D8F AB                             stosw                   ; int 3 ; location 14
00001D90 93                             xchg    ax, bx
00001D91 AB                             stosw                   ; location 16
00001D92 93                             xchg    ax, bx          ; cs:
00001D93 AB                             stosw                   ; int 4 ; location 18
00001D94 89160005                       mov     [0500h], dx
                                        ;mov    ds:500h, dx     ; set print screen & break = 0
00001D98 89160405                       mov     [LSTDRV], dx    ; [0504h]
                                        ;mov    ds:504h, dx     ; clean out last drive spec

                                ; we need to initalize the cs:motorstartup variable from the disk
                                ; parameter table at sec9. the offsets in this table are defined in
                                ; the disk_parms struc in msdskprm.inc. 2 locs

00001D9C A02C05                         mov     al, [SEC9+0Ah] ; 16/10/2022
                                        ;mov    al, ds:52Ch     ; [SEC9+DISK_PARMS.DISK_MOTOR_STRT]
                                                                ; [522h+0Ah]
                                        ; 21/12/2022
                                        ; ds = 0
00001D9F 2EA2[2601]                     mov     [cs:motorstartup], al
00001DA3 2E803E[AF05]FD                 cmp     byte [cs:model_byte], 0FDh ; is this an old rom?
00001DA9 720B                           jb      short no_diddle       ; no
00001DAB C7062B050F02                   mov     word [SEC9+09h], 20Fh
                                        ;mov    word ptr ds:52Bh, 20Fh ; [SEC9+DISK_PARMS.DISK_HEAD_STTL], 0200h+NORMSETTLE
                                                                ; set head settle and motor start on pc-1 pc-2 pc-xt hal0
00001DB1 C6062205DF                     mov     byte [SEC9+0], 0DFh
                                        ;mov    byte ptr ds:522h, 0DFh ; [SEC9+DISK_PARMS.DISK_SPECIFY_1]
                                                                ;  set 1st specify byte       on pc-1 pc-2 pc-xt hal0
                                no_diddle:
00001DB6 CD12                           int     12h             ; MEMORY SIZE -
                                                                ; Return: AX = number of contiguous 1K blocks of memory
00001DB8 B106                           mov     cl, 6
00001DBA D3E0                           shl     ax, cl          ; convert memory size to 16-byte blocks     (segment no.)

                                        ; 21/12/2022
                                        ;pop    cx
                                        ;mov    [cs:drvfat], cx ; save drive to load dos, and fat id

00001DBC 50                             push    ax ; ***        ; save real top       of memory

                                ;M068 - BEGIN
                                ;------ Check if an RPL program is present at TOM and do not tromp over it

                                        ; 21/12/2022
                                        ; ds = 0
                                        ;push   ds
                                        ;push   bx              ; pushes not required but since this
                                        ;                       ; happens to be a last minute change
                                        ;                       ; & since it is only init code.
                                        ;xor    bx, bx
                                        ;mov    ds, bx

                                        ;mov    bx, ds:0BCh     ; [2Fh*4]
00001DBD 8B1EBC00                       mov     bx, [2Fh*4]
                                        ;mov    ds, word ptr ds:0BEh ; [2Fh*4+2]
00001DC1 8E1EBE00                       mov     ds, [2Fh*4+2]
00001DC5 817F035250                     cmp     word [bx+3], 'RP' ; 'RPL'
                                        ;cmp    word ptr [bx+3], 'PR' ; 'RPL'
00001DCA 750F                           jnz     short SkipRPL
00001DCC 807F054C                       cmp     byte [bx+5], 'L'
                                        ;cmp    byte ptr [bx+5], 'L'
00001DD0 7509                           jnz     short SkipRPL
00001DD2 89C2                           mov     dx, ax          ; get TOM into DX
00001DD4 B8064A                         mov     ax, 4A06h       ; (multMULT shl       8) + multMULTRPLTOM
00001DD7 CD2F                           int     2Fh             ; Get new TOM from any RPL
00001DD9 89D0                           mov     ax, dx
                                SkipRPL:
                                        ; 21/12/2022
                                        ;pop    bx
                                        ;pop    ds
                                ;M068 - END
                                        ; 21/12/2022
00001DDB 0E                             push    cs
00001DDC 1F                             pop     ds

00001DDD 83E840                         sub     ax, 64          ; room for fatloc segment. (1 kb buffer)
                                        ; 21/12/2022
00001DE0 A3[021A]                       mov     [fatloc], ax
                                        ;mov    [cs:fatloc], ax       ; location to read fat

```

```
00001DE3 83E840                         sub     ax, 64
00001DE6 A3[041A]                       mov     [init_bootseg], ax ; 21/12/2022
                                        ;mov    [cs:init_bootseg], ax
00001DE9 58                             pop     ax ; ***        ; get back real top of memory for

                                        ; 21/12/2022
00001DEA 59                             pop     cx ; **
00001DEB 890E[FA19]                     mov     [drvfat], cx    ; save drive to load dos, and fat id

                                        ;;mov   dx, 46Dh        ; SYSINIT segment
                                        ;mov    dx, 544h        ; 10/12/2023 (PCDOS 7.1 IBMBIO.COM)
00001DEF BA0405                         mov     dx, SYSINITSEG ; 17/10/2022
00001DF2 8EDA                           mov     ds, dx

                                ; set pointer to resident device driver chain

                                        ; 17/10/2022
00001DF4 C706[7502][2300]               mov     word [DEVICELIST], res_dev_list
                                        ;mov    word [273h], res_dev_list
                                        ;;mov   word ptr ds:273h, offset res_dev_list
                                                                ; [SYSINIT+DEVICE_LIST]
00001DFA 8C0E[7702]                     mov     [DEVICELIST+2], cs
                                        ;mov    [275h], cs
                                        ;;mov   word ptr ds:275h, cs ; [SYSINIT+DEVICE_LIST+2]

00001DFE A3[9402]                       mov     [MEMORYSIZE], ax
                                        ;mov    [292h], ax
                                        ;;mov   ds:292h, ax     ; [SYSINIT+MEMORY_SIZE]

00001E01 FEC1                           inc     cl
00001E03 880E[9802]                     mov     [DEFAULTDRIVE], cl
                                        ;mov    [296h], cl
                                        ;;mov   ds:296h, cl     ; [SYSINIT+DEFAULT_DRIVE]

                                        ;mov    word [CURRENTDOSLOCATION], 0AF8h ; 10/12/2023
00001E07 C706[7302]450A                 mov     word [CURRENTDOSLOCATION], DOSLOADSEG
                                        ;mov    word [271h], 83Fh ; (MSDOS.SYS segment)
                                        ;;mov   word ptr ds:271h, 83Fh ; [SYSINIT+CURRENT_DOS_LOCATION]
                                                                ; dos_load_seg

                                ; important: some old ibm hardware generates spurious int 0F's due to bogus
                                ; printer cards. we initialize this value to point to an iret only if
                                ;
                                ; 1) the original segment points to storage inside valid ram.
                                ;
                                ; 2) the original segment is 0F000:xxxx

                                        ;;;mov  ax, 46Dh        ; SYSINIT segment
                                        ;;mov   ax, 544h        ; 10/12/2023
                                        ;mov    ax, SYSINITSEG ; 17/10/2022
                                        ;mov    es, ax
                                        ; 21/12/2022
00001E0D 8EC2                           mov     es, dx ; SYSINITSEG
00001E0F 31C9                           xor     cx, cx ; 0
00001E11 8ED9                           mov     ds, cx          ; segment 0
                                        ;mov    ax, ds:3Eh      ; [0Fh*4+2]
00001E13 A13E00                         mov     ax, [0Fh*4+2]  ; segment for INT 0Fh
                                        ; 18/10/2022
00001E16 263B06[9402]                   cmp     ax, [es:MEMORYSIZE] ; es:292h
                                        ;cmp    ax, es:292h     ; [ES:SYSINIT+MEMORY_SIZE]  ; (condition 1)
00001E1B 7605                           jbe     short resetintf
00001E1D 3D00F0                         cmp     ax, 0F000h      ; (condition 2)
00001E20 750A                           jnz     short keepintf
                                resetintf:
00001E22 C7063C00[1406]                 mov     word [0Fh*4], intret
                                        ;mov    word ptr ds:3Ch, offset       intret ; [0Fh*4]
00001E28 8C0E3E00                       mov     word [0Fh*4+2], cs
                                        ;mov    word ptr ds:3Eh, cs ; [0Fh*4+2]
                                keepintf:
                                ; end important

                                ; 17/10/2022
                                ; 28/12/2018 - Retro DOS v4.0

                                ; (MSDOS 6.0, MSINIT.ASM, 1991)
                                ;
                                ; we will check if the system has ibm extended keyboard by
                                ; looking at a byte at 40:96. if bit 4 is set, then extended keyboard
                                ; is installed, and we are going to set keyrd_func to 10h, keysts_func to 11h
                                ; for the extended keyboard function. use cx as the temporary register.

                                        ; 21/12/2022
                                        ; ds = 0, cx = 0
                                        ;xor    cx, cx
                                        ;mov    ds, cx

00001E2C 8A0E9604                       mov     cl, [496h]      ; get keyboard flag

                                        ; 21/12/2022
00001E30 0E                             push    cs
00001E31 1F                             pop     ds

00001E32 F6C110                         test    cl, 10h         ; extended keyboard ?
00001E35 740A                           jz      short org_key  ; no, original keyboard

                                        ; 21/12/2022
                                        ; ds = cs
00001E37 C606[7E04]10                   mov     byte [keyrd_func], 10h ; extended keyboard
00001E3C C606[7F04]11                   mov     byte [keysts_func], 11h
                                        ;mov    byte [cs:keyrd_func], 10h ; extended keyboard
                                        ;mov    byte [cs:keysts_func], 11h
                                                                ; change for extended keyboard functions
                                org_key:

                                ; 02/06/2018 - Retro DOS v3.0

                                ;--------------------------------------------------------------
                                ;   will initialize the number of drives
                                ;   after the equipment call (int 11h) bits 6&7 will tell
                                ;   the indications are as follows:
                                ;
                                ;   bits    7       6       drives
                                ;           0       0       1
                                ;           0       1       2
                                ;           1       0       3
                                ;           1       1       4
                                ;--------------------------------------------------------------

                                        ; 21/12/2022
                                        ; ds = cs
                                        ;push   cs
                                        ;pop    ds
                                        ;push   cs
                                        ;pop    es
```

```
                                                call    cmos_clock_read       ; If cmos clock       exists,
                                                                              ; then set the system time according to     that.
                                                                              ; also, reset the cmos clock rate.
                                                ; 18/10/2022
                                                ;mov    word ptr BData_start, offset harddrv ;
                                                                              ; set up pointer to hdrive
                                                ; 02/10/2022
00001E44 C706[0000][4B08]                       mov     word [hdrv_pat], harddrv

00001E4A 58                                     pop     ax ; *                ; number of floppies and FAT ID
00001E4B 30E4                                   xor     ah, ah                ; chuck fat id byte
00001E4D A2[7500]                               mov     [drvmax], al          ; remember which drive is hard disk
00001E50 A2[2501]                               mov     [dsktnum], al         ; and set initial number of drives
00001E53 D1E0                                   shl     ax, 1
00001E55 0106[5E1A]                             add     [last_dskdrv_table], ax

                                                ; 10/12/2023 - Retro DOS v5.0 IO.SYS (IBMBIO.COM)
                                                ; ((MSDOS 6.22 IO.SYS & PCDOS 7.1 IBMBIO.COM))
                                                ; ........
00001E59 1E                                     push    ds
00001E5A B800F0                                 mov     ax, 0F000h
00001E5D 8ED8                                   mov     ds, ax

00001E5F 813EEAFF434F                           cmp     word [0FFEAh], 'CO' ; 'COMPAQ'
00001E65 751F                                   jne     short skip_mode2
00001E67 813EECFF4D50                           cmp     word [0FFECh], 'MP'
00001E6D 7517                                   jne     short skip_mode2
00001E6F 813EEEFF4151                           cmp     word [0FFEEh], 'AQ'
00001E75 750F                                   jne     short skip_mode2

00001E77 B800E4                                 mov     ax, 0E400h     ; get advanced system info (COMPAQ ROMBIOS)
00001E7A CD15                                   int     15h
00001E7C 7208                                   jc      short skip_mode2
                                                ; 10/12/2023
                                                ; PCDOS 7.1 IBMBIO.COM
                                                ;or     bx, 0           ; or bx,40h ; enable mode 2
                                                                        ; (MSDOS 6.0)
                                                                        ; MSDOS 6.22 IO.SYS
00001E7E 83CB40                                 or      bx, 40h         ; enable mode 2 (dual harddisk controller)
00001E81 B880E4                                 mov     ax, 0E480h      ; set advanced system info (COMPAQ ROMBIOS)
00001E84 CD15                                   int     15h
                                        skip_mode2:
00001E86 1F                                     pop     ds
                                                ; ........

00001E87 B280                                   mov     dl, 80h
00001E89 B408                                   mov     ah, 8
00001E8B CD13                                   int     13h             ; DISK - DISK -       GET CURRENT DRIVE PARAMETERS (XT,AT,XT286,CONV,PS)
                                                                        ; DL = drive number
                                                                        ; Return: CF set on error, AH =      status code, BL       = drive type
                                                                        ; DL = number of consecutive drives
                                                                        ; DH = maximum value for head number, ES:DI -> drive parameter
00001E8D 7204                                   jc      short enddrv
00001E8F 8816[5D1A]                             mov     [hnum], dl
                                        enddrv:
                                                ; 21/12/2022
00001E93 0E                                     push    cs
00001E94 07                                     pop     es

                                        ; scan the list of drives to determine their type. we have three flavors of
                                        ; diskette drives:
                                        ;
                                        ;   48tpi drives    we do nothing special for them
                                        ;   96tpi drives    mark the fact that they have changeline support.
                                        ;   3.5"  drives    mark changeline support and small.
                                        ;
                                        ; the following code uses registers for certain values:
                                        ;
                                        ;   dl - physical drive
                                        ;   ds:di - points to current bds
                                        ;   cx - flag bits for bds
                                        ;   dh - form factor for the drive (1 - 48tpi, 2 - 96tpi, 3 - 3.5" medium)

00001E95 30D2                                   xor     dl, dl

                                                ; 21/12/2022
                                                ; ds = cs
                                                ;push   cs
                                                ;pop    ds

00001E97 C606[2C01]09                           mov     byte [eot], 9
00001E9C BF[1901]                               mov     di, start_bds   ; if we are faking floppy drives we need
                                                                        ; to set aside two bdss       for the two fake floppy       drives

                                        ; 02/10/2022 - Retro DOS v4.0 (MSDOS 5.0 IO.SYS)
                                        ; 28/12/2018 - Retro DOS v4.0 (MSDOS 6.0, MSINIT.ASM)

                                        ; check to see if we are faking floppy drives. if not we don't
                                        ; do anything special. if we are faking floppy drives we need
                                        ; to set aside two bdss for the two fake floppy drives. we
                                        ; don't need to initalise any fields though. so starting at start_bds
                                        ; use the link field in the bds structure to go to the second bds
                                        ; in the list and initalise it's link field to -1 to set the end of
                                        ; the list. then jump to the routine at dohard to allocate/initialise
                                        ; the bds for harddrives.

00001E9F 803E[111A]01                               cmp     byte [fakefloppydrv], 1
00001EA4 750B                                       jnz     short loop_drive
00001EA6 8B3D                                       mov     di, [di]        ; [di+BDS.link]
                                                                            ; di <- first bds link
00001EA8 8B3D                                       mov     di, [di]        ; [di+BDS.link]
                                                                            ; di <- second bds link
00001EAA C705FFFF                                   mov     word [di], 0FFFFh ; -1 ; set end of link
00001EAE E98801                                     jmp     dohard          ; allocate/initialise bds for harddrives
                                        ;-----------------------------------------------------------------------------

                                        loop_drive:
00001EB1 3A16[7500]                                 cmp     dl, [drvmax]
00001EB5 7203                                       jb      short got_more
00001EB7 E97B01                                     jmp     done_drives
                                        ;-----------------------------------------------------------------------------

                                        got_more:
                                                    ; 10/12/2023
                                                    ;xor    cx, cx          ; zero all flags
00001EBA 8B3D                                       mov     di, [di]        ; [di+BDS.link]
                                                                            ; get next bds
                                                    ; ........
                                                    ; 10/12/2023 - Retro DOS v5.0
                                                    ; (PCDOS 7.1 IBMBIO.COM BIOSDATA:2046h)
00001EBC 83FFFF                                     cmp     di, 0FFFFh      ; end of link ?
00001EBF 7516                                       jne     short not_last_bds
00001EC1 88D0                                       mov     al, dl          ; drive number (0 based)
```

```
00001EC3 98                             cbw
00001EC4 01C0                           add     ax, ax
00001EC6 05[3C05]                       add     ax, dskdrvs
00001EC9 A3[5E1A]                       mov     [last_dskdrv_table], ax
00001ECC 8B3E[601A]                     mov     di, [end_of_bdss]
00001ED0 E8020B                         call    xinstall_bds
00001ED3 FE0E[7500]                     dec     byte [drvmax]
                                not_last_bds:
                                        ; ........

00001ED7 B600                           mov     dh, 0           ; ff48tpi
                                                                ; set form factor to 48      tpi
00001ED9 C606[0E1A]28                   mov     byte [num_cyln], 40 ; 40 tracks per side

                                        ; 21/12/2022
                                        ;push   ds
00001EDE 57                             push    di
00001EDF 52                             push    dx
                                        ;push   cx ; not necessary (10/12/2023)
00001EE0 06                             push    es ; es=cs=ds ; 21/12/2022

                                        ; ............
                                        ; 10/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                        ;xor    bx, bx
                                        ;xor    cx, cx
00001EE1 52                             push    dx  ; dl = drive number

00001EE2 B408                           mov     ah, 8
00001EE4 CD13                           int     13h             ; DISK - DISK -       GET CURRENT DRIVE PARAMETERS (XT,AT,XT286,CONV,PS)
                                                                ; DL = drive number
                                                                ; Return: CF set on error, AH =      status code, BL       = drive type
                                                                ; DL = number of consecutive drives
                                                                ; DH = maximum value for head number, ES:DI -> drive parameter
                                        ;jc     short noparmsfromrom
                                        ; 10/12/2023
00001EE6 58                             pop     ax  ; al = drive number
00001EE7 7303                           jnc     short chk_drv_type
00001EE9 E9E600                         jmp     noparmsfromrom

                                chk_drv_type:
                                        ; 10/12/2023
                                        ; ch = low eight bits of maximum cylinder number
                                        ; cl = maximum sector number (bits 5-0)
                                        ;      high two bits of maximum cylinder number (bits 7-6)
                                        ;
00001EEC 80FB10                         cmp     bl, 10h         ; ATAPI Removable Media Device
00001EEF 7554                           jne     short not_atapi_removable

                                        ; save ds:si
00001EF1 1E                             push    ds
                                        ;push   si      ; not necessary (10/12/2023)

00001EF2 88C2                           mov     dl, al
00001EF4 83EC1A                         sub     sp, 26
00001EF7 31C0                           xor     ax, ax ; 0
00001EF9 50                             push    ax
00001EFA B81E00                         mov     ax, 30
00001EFD 50                             push    ax
00001EFE 89E6                           mov     si, sp          ; DS:SI = segment:offset pointer to Result Buffer
00001F00 16                             push    ss
00001F01 1F                             pop     ds
00001F02 B448                           mov     ah, 48h
00001F04 CD13                           int     13h             ; DISK - IBM/MS Extension
                                                                ; GET DRIVE PARAMETERS (DL - drive, DS:SI - buffer)
00001F06 7239                           jc      short ext_gdp_err
00001F08 8B4408                         mov     ax, [si+8]      ; physical number of heads
00001F0B A3[0C1A]                       mov     [num_heads], ax
00001F0E 8B4404                         mov     ax, [si+4]      ; physical number of cylinders
00001F11 A3[0E1A]                       mov     [num_cyln], ax
00001F14 8A440C                         mov     al, [si+0Ch]    ; physical number of sectors per track
00001F17 A2[101A]                       mov     [sec_trk], al
00001F1A 3A06[2C01]                     cmp     al, [eot]
00001F1E 7603                           jbe     short _eotok
00001F20 A2[2C01]                       mov     [eot], al

                                _eotok:                 ; 10/12/2023
                                        ;xor    al, al
00001F23 31C9                           xor     cx, cx ; 0
00001F25 F6440210                       test    byte [si+2], 10h ; information flags
                                                                ; bit 4 = Device has change line support
00001F29 7403                           jz      short not_chgline_sup
                                        ;or     al, 2           ; change line support
00001F2B 80C902                         or      cl, 2
                                not_chgline_sup:
00001F2E 83C41E                         add     sp, 30
                                        ;pop    si      ; (10/12/2023)
00001F31 1F                             pop     ds
                                        ;
00001F32 07                             pop     es      ; es=cs=ds (21/12/2022)
                                        ;pop    cx      ; (10/12/2023)
00001F33 5A                             pop     dx
00001F34 5F                             pop     di
                                        ;pop    ds      ; (21/12/2022)

                                        ; 10/12/2023
00001F35 F6C102                         test    cl, 2
                                        ;test   al, 2
                                        ;jz     short gotother_j
00001F38 7450                           jz      short gotother
                                        ;or     cl, al
00001F3A C606[7700]01                   mov     byte [fhave96], 1 ; Device has change line support
                                gotother_j:
00001F3F EB49                           jmp     short gotother
                                ext_gdp_err:
00001F41 83C41E                         add     sp, 30
                                        ;pop    si      ; (10/12/2023)
00001F44 1F                             pop     ds

                                        ; 10/12/2023
                                not_atapi_removable:
                                        ; ...........

                                ; if cmos is bad, it gives es,ax,bx,cx,dh,di=0. cy=0.
                                ; in this case, we are going to put bogus informations to bds table.
                                ; we are going to set ch=39,cl=9,dh=1 to avoid divide overflow when
                                ; they are calculated at the later time. this is just for the diagnostic
                                ; diskette which need msbio,msdos to boot up before it sets cmos.
                                ; this should only happen with drive b.

00001F45 80FD00                         cmp     ch, 0           ; if ch=0, then       cl,dh=0 too.
00001F48 7505                           jnz     short pfr_ok

                                        ;mov    ch, 39          ; rom gave wrong info.
                                        ;mov    cl, 9           ; let's default to 360k.
```

```
                                        ; 21/12/2022
00001F4A B90927                         mov     cx, 2709h
00001F4D B601                           mov     dh, 1
                                pfr_ok:
                                        ;inc    dh              ; make number of heads 1-based
                                        ;mov    [num_heads], dh       ; save parms returned by rom
                                        ; 10/12/2023
00001F4F 86F2                           xchg    dl, dh
00001F51 30F6                           xor     dh, dh
00001F53 42                             inc     dx              ; make number of heads 1-based
00001F54 8916[0C1A]                     mov     [num_heads], dx

                                        ;inc    ch              ; make number of cylinders 1-based
                                        ;and    cl, 3Fh
                                        ;mov    [sec_trk], cl
                                        ;mov    [num_cyln], ch ; assume less than 256 cylinders!!
                                        ; 10/12/2023
00001F58 88CA                           mov     dl, cl
00001F5A 80E23F                         and     dl, 3Fh
00001F5D 8816[101A]                     mov     [sec_trk], dl
00001F61 86E9                           xchg    cl, ch
00001F63 D0C5                           rol     ch, 1
00001F65 D0C5                           rol     ch, 1
00001F67 80E503                         and     ch, 3
00001F6A 41                             inc     cx              ; make number of cylinders 1-based
00001F6B 890E[0E1A]                     mov     [num_cyln], cx

                                ; make sure that eot contains the max number of sec/trk in system of floppies

                                        ;mov    cl, [sec_trk] ; 10/12/2023
                                        ;cmp    cl, [eot]       ; may set carry
                                        ;;jbe   short eot_ok
                                        ;; 09/12/2022
                                        ;;jne   short eotok  ; wrong ! 14/08/2023
                                        ;; 14/08/2023
                                        ;jbe    short eotok
                                        ;mov    [eot], cl
                                        ; 10/12/2023
00001F6F 3A16[2C01]                     cmp     dl, [eot] ; dl = [sec_trk]
00001F73 7604                           jbe     short eotok
00001F75 8816[2C01]                     mov     [eot], dl
                                ;eot_ok:
                                eotok:
                                        ; 10/12/2023
                                        ; !!!
                                        ; (following pops are moved to 'chk_changeline' procedure)
                                        ;pop    es ; es=cs=ds ; 21/12/2022
                                        ;;pop   cx      ; (10/12/2023)
                                        ;pop    dx
                                        ;pop    di

                                        ; 21/12/2022
                                        ;pop    ds

                                ; Check     for presence of       changeline

                                ; 10/12/2023
                                %if 0
                                        ; 10/12/2023
                                        ;xor    cx, cx  ; 0
                                        ;push   cx
                                        push    dx

                                        mov     ah, 15h
                                        int     13h             ; DISK - DISK -       GET TYPE (AT,XT2,XT286,CONV,PS)
                                                                ; DL = drive ID
                                                                ; Return: CF set on error, AH =      disk type (3 = hard drive)
                                                                ; CX:DX= number of sectors on the media
                                        ; 10/12/2023
                                        pop     dx
                                        ;pop    cx
                                        mov     cx, 0 ; 12/12/2023
                                        jc      short changeline_done
                                        cmp     ah, 2           ; check for presence of       changeline
                                        jnz     short changeline_done

                                ; we have a drive with change line support.

                                        or      cl, 2           ; fchangeline
                                                                ; signal type
                                        mov     byte [fhave96], 1 ; remember that we have 96tpi disks
                                %endif
                                        ; 10/12/2023
00001F79 E83800                         call    chk_changeline
                                        ;jc     short changeline_done

                                ; we now try to set up the form factor for the types of media that we know
                                ; and can recognise. for the rest, we set the form factor as "other".

                                changeline_done:
00001F7C 803E[0E1A]28                           cmp     byte [num_cyln], 40
00001F81 750B                                   jnz     short try_80
00001F83 803E[101A]09                           cmp     byte [sec_trk], 9
00001F88 765F                                   jbe     short nextdrive
                                gotother:
                                                ; 10/12/2023
                                                ; ch = 0, cl = 2 or 0

00001F8A B607                                   mov     dh, 7           ; ffOther
                                                                        ; we have a "strange" medium
00001F8C EB5B                                   jmp     short nextdrive
                                ;-----------------------------------------------------------------------------

                                ; 80 cylinders and 9 sectors/track => 720 kb device
                                ; 80 cylinders and 15 sec/trk => 96 tpi medium

                                try_80:
00001F8E 803E[0E1A]50                           cmp     byte [num_cyln], 80
00001F93 75F5                                   jnz     short gotother
00001F95 B609                                   mov     dh, 9           ; ff288
                                                                        ; assume 2.88 MB drive
00001F97 803E[101A]24                           cmp     byte [sec_trk], 36 ; is it   ?
00001F9C 744B                                   jz      short nextdrive       ; yeah, go update

                                                ; 12/05/2019 (ff144 type will not be used -compatibility problem-)
                                                ; 08/01/2018 - Retro DOS v4.0 feature only ! for 1.44MB diskettes
                                                ;mov    dh, ff144
                                                ;cmp    byte [sec_trk], 18
                                                ;je     short nextdrive

00001F9E 803E[101A]0F                           cmp     byte [sec_trk], 15
00001FA3 740B                                   jz      short got96

00001FA5 803E[101A]09                           cmp     byte [sec_trk], 9
```

```
7319 00001FAA 75DE                              jnz     short gotother
7320
7321 00001FAC B602                              mov     dh, 2           ; ffSmall
7322 00001FAE EB39                              jmp     short nextdrive
7323                                    ; ---------------------------------------------------------------------------
7324
7325                                    got96:
7326 00001FB0 B601                              mov     dh, 1           ; ff96tpi
7327 00001FB2 EB35                              jmp     short nextdrive
7328
7329                                    ; ---------------------------------------------------------------------------
7330
7331                                            ; 10/12/2023 - Retro DOS v5.0 IO.SYS/IBMBIO.COM
7332                                            ; check change line feature (and set fhave96 if there is)
7333                                            ; (common procedure for 'eotok:' and 'noparmsfromrom:')
7334                                    chk_changeline:
7335 00001FB4 59                               pop     cx ; near call return address
7336
7337                                            ; (pop es, dx, di for 'eotok' and 'noparmsfromrom' procs)
7338 00001FB5 07                               pop     es ; es=cs=ds ; 21/12/2022
7339                                            ;pop    cx      ; (10/12/2023)
7340 00001FB6 5A                               pop     dx
7341 00001FB7 5F                               pop     di ; BDS address/offset
7342
7343 00001FB8 51                               push    cx ; near call return address
7344
7345                                            ;xor    cx, cx ; 0
7346                                            ;push   cx
7347 00001FB9 52                               push    dx
7348
7349 00001FBA B415                              mov     ah, 15h
7350 00001FBC CD13                              int     13h             ; DISK - DISK -       GET TYPE (AT,XT2,XT286,CONV,PS)
7351                                                                    ; DL = drive ID
7352                                                                    ; Return: CF set on error, AH =      disk type (3 = hard drive)
7353                                                                    ; CX:DX= number of sectors on the media
7354 00001FBE 5A                               pop     dx
7355                                            ;pop    cx
7356 00001FBF B90000                            mov     cx, 0
7357 00001FC2 720D                              jc      short chk_chgl_1
7358
7359 00001FC4 80FC02                            cmp     ah, 2           ; is there changeline?
7360 00001FC7 7508                              jne     short chk_chgl_2 ; *
7361
7362 00001FC9 80C902                            or      cl, 2
7363                                            ;or     cl, ah ; 2
7364 00001FCC C606[7700]01                      mov     byte [fhave96], 1 ; fchangeline
7365                                            ; cf = 0
7366                                    chk_chgl_1:
7367                                    chk_chgl_2:
7368 00001FD1 C3                                retn
7369
7370                                    ;chk_chgl_2:       ; *
7371                                    ;           ; 10/12/2023
7372                                    ;           ; ah = 1 ; harddisk type (ah = 3) return not possible here for floppies
7373                                    ;           ;stc
7374                                    ;           ; cf = 1
7375                                    ;           retn
7376
7377                                    ; ---------------------------------------------------------------------------
7378
7379                                    ; we have an old rom, so we either have a 48tpi or 96tpi drive. if the drive
7380                                    ; has changeline, we assume it is a 96tpi, otherwise we treat it as a 48tpi.
7381
7382                                    noparmsfromrom:
7383                                            ; 10/12/2023
7384                                            ; !!!
7385                                            ; (following pops are moved to 'chk_changeline' procedure)
7386                                            ;pop    es ; es=cs=ds ; 21/12/2022
7387                                            ;;pop   cx      ; (10/12/2023)
7388                                            ;pop    dx
7389                                            ;pop    di
7390
7391                                            ; 21/12/2022
7392                                            ;pop    ds
7393                                    ; 10/12/2023
7394                                    %if 0
7395                                            ; 10/12/2023
7396                                            ;xor    cx, cx ; 0
7397                                            ;push   cx
7398                                            push    dx
7399
7400
7401                                            mov     ah, 15h
7402                                            int     13h             ; DISK - DISK -       GET TYPE (AT,XT2,XT286,CONV,PS)
7403                                                                    ; DL = drive ID
7404                                                                    ; Return: CF set on error, AH =      disk type (3 = hard drive)
7405                                                                    ; CX:DX= number of sectors on the media
7406                                            ; 10/12/2023
7407                                            pop     dx
7408                                            ;pop    cx
7409                                            mov     cx, 0 ; 12/12/2023
7410                                            jc      short nextdrive
7411
7412                                            cmp     ah, 2           ; is there changeline?
7413                                            jnz     short nextdrive
7414
7415                                            or      cl, 2
7416                                            mov     byte [fhave96], 1 ; fchangeline
7417                                    %endif
7418                                            ; 10/12/2023
7419 00001FD2 E8DFFF                           call    chk_changeline
7420 00001FD5 7212                             jc      short nextdrive
7421
7422                                            ; change line support, [fhave96] = 1
7423
7424 00001FD7 C606[0E1A]50                      mov     byte [num_cyln], 80
7425 00001FDC B601                              mov     dh, 1           ; ff96tpi
7426 00001FDE B00F                              mov     al, 15
7427 00001FE0 3A06[2C01]                        cmp     al, [eot]
7428 00001FE4 7603                              jbe     short nextdrive
7429 00001FE6 A2[2C01]                          mov     [eot], al
7430                                    ; ---------------------------------------------------------------------------
7431
7432                                    nextdrive:
7433                                            ; 10/12/2023
7434                                            ; ch = 0, cl = 2 or 0
7435
7436 00001FE9 80C920                            or      cl, 20h         ; fi_own_physical
7437                                                                    ; set this true       for all drives
7438 00001FEC 88D7                              mov     bh, dl          ; save int13 drive number
7439
7440                                    ; we need to do special things if we have a single drive system and are setting
7441                                    ; up a logical drive. it needs to have the same int13 drive number as its
7442                                    ; counterpart, but the next drive letter. also reset ownership flag.
```

```
                                ; we detect the presence of this situation by examining the flag single for the
                                ; value 2.
00001FEE 803E[7800]02                   cmp     byte [single], 2
00001FF3 7505                           jnz     short not_special
00001FF5 FECF                           dec     bh              ; int13 drive number same for logical drive
00001FF7 80F120                         xor     cl, 20h         ; reset ownership flag for logical drive
                                not_special:

                                ; the values that we put in for BDS_RBPB.BPB_HEADS and
                                ; BDS_RBPB.BPB_SECTORSPERTRACK will only remain if the
                                ; form factor is of type "ffother".

                                        ;xor    ax, ax          ; fill BDS for drive
                                        ;mov    al, [num_heads]
                                        ; 10/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM) ; *
00001FFA A1[0C1A]                       mov     ax, [num_heads]
                                        ;mov    [di+36h], ax    ; [di+BDS.rheads]
00001FFD 894552                         mov     [di+52h], ax    ; [di+BDS.rheads] ; *
00002000 31C0                           xor     ax, ax          ; *
00002002 A0[101A]                       mov     al, [sec_trk]
                                        ;mov    [di+34h], ax    ; [di+BDS.rsecpertrack]
00002005 894550                         mov     [di+50h], ax    ; [di+BDS.rsecpertrack] ; *
                                        ;mov    [di+23h], cx    ; [di+BDS.flags]
00002008 894D3F                         mov     [di+3Fh], cx    ; [di+BDS.flags] ; *
                                        ;mov    [di+22h], dh    ; [di+BDS.formfactor]
0000200B 88753E                         mov     [di+3Eh], dh    ; [di+BDS.formfactor] ; *
0000200E 885505                         mov     [di+5], dl      ; [di+BDS.drivelet]
00002011 887D04                         mov     [di+4], bh      ; [di+BDS.drivenum]
                                        ;mov    bl, [num_cyln]
                                        ;mov    [di+25h], bl    ; [di+BDS.cylinders]
                                        ; 10/12/2023
00002014 A1[0E1A]                       mov     ax, [num_cyln]
00002017 894541                         mov     [di+41h], ax    ; [di+BDS.cylinders] ; *

0000201A 803E[7800]01                   cmp     byte [single], 1 ; Special case for single drive system
0000201F 7510                           jnz     short no_single
                                        ;mov    byte [single], 2 ; Don't forget we have
                                                                 ; single drive system
                                        ; 10/12/2023
00002021 FE06[7800]                     inc     byte [single]  ; [single] = 2
                                        ; 18/12/2022
00002025 80C910                         or      cl, 10h
                                        ;or     cx, 10h         ; fi_am_mult
                                                                ; set that this       is one of several drives
                                        ;or     [di+23h], cx    ; [di+BDS.flags]
00002028 094D3F                         or      [di+3Fh], cx    ; [di+BDS.flags] ; *
                                                                ; save flags
0000202B 8B3D                           mov     di, [di]        ; [di+BDS.link]
                                                                ; move to next BDS in list
0000202D FEC2                           inc     dl              ; add a number
0000202F EBB8                           jmp     short nextdrive       ; Use same info       for BDS as previous
                                ; ---------------------------------------------------------------------------

                                no_single:
                                        ;inc    dl
                                        ; 18/12/2022
00002031 42                             inc     dx
00002032 E97CFE                         jmp     loop_drive
                                ; ---------------------------------------------------------------------------

                                        ; 11/12/2023 - Retro DOS v5.0 IO.SYS/IBMBIO.COM
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:21E8h)
                                done_drives:
                                        ;mov    word [di+BDS.link], -1
00002035 C705FFFF                       mov     word [di], -1  ; set link to null

                                ; set up all the hard drives in   the system

                                dohard:
00002039 8A36[5D1A]                     mov     dh, [hnum]
0000203D 08F6                           or      dh, dh          ; done if no hardfiles
0000203F 7459                           jz      short static_configure
00002041 B280                           mov     dl, 80h
                                dohard1:
00002043 52                             push    dx
00002044 8B3E[601A]                     mov     di, [end_of_bdss]
00002048 8A1E[7500]                     mov     bl, [drvmax]
0000204C B700                           mov     bh, 0           ; first primary partition (or active)
0000204E E89902                         call    sethard
00002051 7208                           jc      short hardfile_err
00002053 E86A09                         call    dmax_check      ; error if already 26 drives
00002056 7303                           jnb     short hardfile_err
00002058 E87A09                         call    xinstall_bds    ; insert new bds into linked list
                                hardfile_err:
0000205B 5A                             pop     dx
                                        ;inc    dl              ; next hard drive
                                        ; 12/12/2023
0000205C 42                             inc     dx
0000205D FECE                           dec     dh
0000205F 75E2                           jnz     short dohard1

                                ; end of physical drive   initialization

                                ; *** do not change the position of the following statement.
                                ; *** domini routine will use [drvmax] value for the start of the logical
                                ; *** drive number of mini disk(s).

00002061 E8CF07                         call    domini          ; for setting up mini disks, if found

                                ; -- begin added section

00002064 8A36[5D1A]                     mov     dh, [hnum]      ; we already know this is >0
00002068 B280                           mov     dl, 80h
                                dohardx1:
0000206A B701                           mov     bh, 1           ; do all subsequent primary partitions
                                dohardx2:
0000206C 52                             push    dx
0000206D 53                             push    bx
0000206E 8B3E[601A]                     mov     di, [end_of_bdss]
00002072 8A1E[7500]                     mov     bl, [drvmax]
00002076 E87102                         call    sethard
00002079 720E                           jc      short dohardx4 ; move to next hardfile       if error
0000207B E84209                         call    dmax_check      ; make sure <=26 drives
0000207E 7309                           jnb     short dohardx4 ; skip if error
00002080 E85209                         call    xinstall_bds    ; insert new bds into linked list
00002083 5B                             pop     bx              ; get partition       number
00002084 5A                             pop     dx              ; restore physical drive counts
00002085 FEC7                           inc     bh
00002087 EBE3                           jmp     short dohardx2 ; keep looping until we       fail
                                ; ---------------------------------------------------------------------------

                                dohardx4:
00002089 5B                             pop     bx              ; unjunk partition number from stack
0000208A 5A                             pop     dx              ; restore physical drive counts
```

```
                                        ;inc    dl              ; next hard drive
                                        ; 12/12/2023
0000208B 42                             inc     dx
0000208C FECE                           dec     dh
0000208E 75DA                           jnz     short dohardx1

                                ; -- end changed section

                                ;********************************************************************************
                                ; if more than 2 diskette drives on the system, then it is necessary to remap
                                ; the bds chain to adjust the logical drive num (drive letter) with greater
                                ; than two diskette drives
                                ;
                                ; new scheme:      if more than 2 disktte drives, first map the bds structure
                                ;           as usual and then rescan the bds chain to adjust the  drive
                                ;           letters. to do this, scan for disk drives and assign logical
                                ;           drive number starting from 2 and then rescan diskette drives
                                ;           and assign next to the last logical drive number of last disk
                                ;           drive to the 3rd and 4th diskette drives.
                                ;********************************************************************************

00002090 803E[2501]02                   cmp     byte [dsktnum], 2 ; >2 diskette drives
                                        ;jbe    short static_configure ; no - no need for remapping
00002095 7603                           jbe     short no_remap
00002097 E8D401                         call    remap           ; remap bds chain to adjust driver letters
                                no_remap:

                                ; End of drive initialization.

                                ; ---------------------------------------------------------------------------

                                ;we now decide, based on the configurations available so far, what
                                ;code or data we need to keep as a stay resident code. the following table
                                ;shows the configurations under consideration. they are listed in the order
                                ;of their current position memory.
                                ;
                                ;configuration will be done in two ways:
                                ;
                                ;first, we are going to set "static configuration". static configuration will
                                ;consider from basic configuration to endof96tpi configuration. the result
                                ;of static configuration will be the address the dynamic configuration will
                                ;use to start with.
                                ;
                                ;secondly, "dynamic cofiguration" will be performed. dynamic configuration
                                ;involves possible relocation of code or data. dynamic configuration routine
                                ;will take care of bdsm tables and at rom fix module thru k09 suspend/resume
                                ;code individually. after these operation, [dosdatasg] will be set.
                                ;this will be the place sysinit routine will relocate msdos module for good.

                                ; -- begin changed section
                                ;
                                ;   1.      basic configuration for msbio (endfloppy)
                                ;   2.   end96tpi  ; a system that supports "change line error"
                                ;   3.      end of bdss   ; end of bdss for hard disks
                                ;   4.      endatrom      ;some of at rom fix module.
                                ;   5.      endcmosclockset;supporting program for cmos clock write.
                                ;   6.      endk09        ;k09 cmos clock module to handle suspend/resume operation.
                                ;

                                ; 02/10/2022 - Retro DOS v4.0 (MSDOS v5.0 IO.SYS)

                                static_configure:
0000209A 8B3E[601A]                     mov     di, [end_of_bdss]
0000209E 81FF[4C08]                     cmp     di, bdss        ; 19/10/2022
                                        ;cmp    di, offset bdss         ; did we allocate any hard drive bdss?
000020A2 750D                           jnz     short dynamic_configure       ; that's the end, then
                                        ; 18/10/2022
000020A4 BF[4C08]                       mov     di, end96tpi
                                        ;mov    di, offset harddrv ; end96tpi
                                                                ; keep everything up to       end96tpi
000020A7 803E[7700]00                   cmp     byte [fhave96], 0
000020AC 7503                           jnz     short dynamic_configure

000020AE BF[3808]                       mov     di, endfloppy
                                dynamic_configure:
                                        ; 20/12/2022
                                        ;push   cs
                                        ;pop    es

                                        ; 10/12/2023
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2268h)
                                        ; (MSDOS 6.22 IO.SYS - BIOSDATA:1C34h)
000020B1 FC                             cld     ; clear direction flag is not necessary here !?
                                                ; because there will not be a running program
                                                ; which will set direction flag as backward (std)

                                ; -- end changed section
                                        ; 21/12/2022
                                        ; ds = cs <> es
                                        ; ss = 0
                                        ; sp = 700h

                                        ; 13/12/2023
000020B2 BE00F0                         mov     si, 0F000h
000020B5 8EC6                           mov     es, si          ; ES -> ROM BIOS segment

000020B7 803E[AF05]FC                   cmp     byte [model_byte], 0FCh ; AT ?
                                        ;jnz    short checkcmosclock
                                        ; 10/12/2023
000020BC 751E                           jnz     short checkcompaqbug ; no
000020BE 803E[5D1A]00                   cmp     byte [hnum], 0 ; No hard file?
                                        ;jz     short checkcmosclock
000020C3 7417                           jz      short checkcompaqbug
000020C5 97                             xchg    ax, di          ; save allocation pointer in ax
                                        ; 13/12/2023
                                        ;mov    si, 0F000h
                                        ;mov    es, si          ; ES -> ROM BIOS segment
000020C6 BE[661A]                       mov     si, bios_date   ; "01/10/84"
000020C9 BFF5FF                         mov     di, 0FFF5h      ; ROM BIOS string is at       F000:FFF5
000020CC B90900                         mov     cx, 9           ; bdate_l
                                                                ; Only patch ROM for bios 01/10/84
000020CF F3A6                           repe cmpsb              ; check for date + zero       on end
000020D1 97                             xchg    ax, di          ; restore allocation pointer

                                ; M015 -- begin changes

                                        ;jnz    short checkcmosclock
                                        ; 02/10/2022
000020D2 7508                           jnz     short checkcompaqbug

                                ; install at rom fix

                                        ; 19/10/2022
```

```
                                        ;mov    cx, offset endatrom
000020D4 B9[2018]                       mov     cx, endatrom
                                        ;mov    si, offset ibm_disk_io
000020D7 BE[F216]                       mov     si, ibm_disk_io
000020DA EB46                           jmp     short install_int13_patch
                                ; ---------------------------------------------------------------------------

                                ; M065 -- begin changes
                                ;
                                ; On certain systems with Western Digital disk controllers, the
                                ; following detection scheme caused an unpredictable and serious
                                ; failure. In particular, they've implemented a nonstandard
                                ; Int13(ah=16h) which reconfigures the hard drive, depending on
                                ; what happens to be at es:[bx] and other memory locations indexed
                                ; off of it.
                                ;
                                ; Compaq was unable to tell us exactly which kind of systems have
                                ; the bug, except that they guarantee that the bug was fixed in
                                ; ROM BIOSs dated 08/04/86 and later. We'll check for the COMPAQ
                                ; string, and then look for date codes before 08/04/86 to decide
                                ; when to install the hook.

                                ;checkcmosclock:
                                ; 02/10/2022
                                checkcompaqbug:
                                        ; 21/12/2022
                                        ; es = 0F000h
                                        ;mov    ax, 0F000h      ; point to ROM BIOS
                                        ;mov    es, ax

                                        ; 19/10/2022
000020DC 26813EEAFF434F                 cmp     word [es:0FFEAh], 'CO'
                                        ;cmp    word ptr es:0FFEAh, 'OC' ; look for COMPAQ
000020E3 754B                           jnz     short not_compaq_patch
000020E5 26813EECFF4D50                 cmp     word [es:0FFECh], 'MP'
                                        ;cmp    word ptr es:0FFECh, 'PM'
000020EC 7542                           jnz     short not_compaq_patch
000020EE 26813EEEFF4151                 cmp     word [es:0FFEEh], 'AQ'
                                        ;cmp    word ptr es:0FFEEh, 'QA'
000020F5 7539                           jnz     short not_compaq_patch

                                ; We're running on a COMPAQ. Now look at the date code.

000020F7 26A1FBFF                       mov     ax, [es:0FFFBh]       ; get year
000020FB 86C4                           xchg    ah, al

                                ; 11/12/2023
                                ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:22B9h)
                                %if 0
                                        cmp     ax, 3836h       ; '68' (NASM syntax) (('86' in MASM syntax))
                                        ja      short checkk09
                                        jz      short chkcompaqbug1
                                        cmp     ax, 3739h       ; '97'
                                        jbe     short not_compaq_patch
                                        stc
                                chkcompaqbug1:
                                        jb      short do_compaq_patch
                                        mov     ax, [es:0FFF5h]
                                        xchg    ah, al
                                        cmp     ax, 3038h       ; '80'
                                        ja      short not_compaq_patch
                                        jb      short do_compaq_patch
                                        mov     ax, [es:0FFF8h]
                                        xchg    ah, al
                                        cmp     ax, 3034h       ; '40'
                                        jnb     short not_compaq_patch
                                do_compaq_patch:
                                %endif
                                        ; 11/12/2023
                                        ; (MSDOS 6.22 IO.SYS - BIOSDATA:1C85h)

000020FD 3D3638                         cmp     ax, 3836h ; 02/10/2022 (NASM syntax)
                                        ;cmp    ax, '86'        ; 3836h
                                                                ; is it 86?
00002100 772E                           ja      short not_compaq_patch
00002102 7218                           jb      short do_compaq_patch
00002104 26A1F5FF                       mov     ax, [es:0FFF5h]       ; get month
00002108 86C4                           xchg    ah, al
0000210A 3D3830                         cmp     ax, 3038h ; 02/10/2022 (NASM syntax)
                                        ;cmp    ax, '08'        ; 3038h
                                                                ; is it 08?
0000210D 7721                           ja      short not_compaq_patch
0000210F 720B                           jb      short do_compaq_patch
00002111 26A1F8FF                       mov     ax, [es:0FFF8h]       ; get day
00002115 86C4                           xchg    ah, al
00002117 3D3430                         cmp     ax, 3034h ; 02/10/2022 (NASM syntax)
                                        ;cmp    ax, '04'        ; 3034h
                                                                ; is it 04?
0000211A 7314                           jnb     short not_compaq_patch

                                do_compaq_patch:
0000211C B9[3D18]                       mov     cx, end_compaq_i13hook
                                        ;mov    si, endatrom
                                        ; 11/12/2023
0000211F BE[2018]                       mov     si, compaq_disk_io ; endatrom

                                install_int13_patch:
00002122 0E                             push    cs
00002123 07                             pop     es
                                        ; 18/10/2022
00002124 893E[B400]                     mov     [Orig13], di   ; set new rom bios int 13 vector
00002128 8C0E[B600]                     mov     [Orig13+2], cs
0000212C 29F1                           sub     cx, si         ; size of rom fix module
0000212E F3A4                           rep movsb              ; relocate it

                                ; M065 -- end changes

                                ; ---------------------------------------------------------------------------
                                not_compaq_patch:               ; M065
                                        ; 17/10/2022
                                checkcmosclock:
                                        ; 18/10/2022
00002130 0E                             push    cs
00002131 07                             pop     es

                                        ; 21/12/2022
                                        ; ds = cs = es
                                        ; ss = 0
                                        ; sp = 700h

                                ; 09/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                %if 0
                                        cmp     byte [havecmosclock], 1 ; cmos clock exists?
                                        jnz     short checkk09 ; no
```

```
                                        mov     word [daycnttoday], di
                                        ;mov    word ptr ds:daycnttoday, di ; set the address for mschar
                                        mov     cx, 209 ; enddaycnttoday - daycnt_to_day
                                        mov     si, daycnt_to_day
                                        rep movsb
                                        mov     word [bintobcd], di
                                        ;mov    word ptr ds:bintobcd, di ; set the address for msclock
                                                                         ; let original segment stay
                                        ;mov    cx, 11  ; endcmosclockset - bin_to_bcd
                                        ; 08/08/2023
                                        mov     cl, 11
                                        mov     si, bin_to_bcd
                                        rep movsb
                                %endif

                                ; 09/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:22F4h
                                        ;push   cs
                                        ;pop    es
                                checkk09:
00002132 57                             push    di ; ? ; save ? ; 21/12/2022

                                ; 13/12/2023 - Retro DOS v4.2 IO.SYS
                                ; (MSDOS 6.22 IO.SYS - BIOSDATA:1CDAh)
                                %if 0

                                        mov     ax, 4101h       ; wait for bh=es:[di]
                                        mov     bl, 1           ; wait for 1 clock tick
                                        mov     bh, [es:di]
                                        stc                     ; Assume we will fail
                                        int     15h             ; SYSTEM - WAIT       ON EXTERNAL EVENT (CONVERTIBLE)
                                                                ; AL = condition type, BH = condition compare or mask value
                                                                ; BL = timeout value times 55 milliseconds, 00h     means no timeout
                                                                ; DX = I/O port       address if AL bit 4 set
                                                                ; 11/12/2023
                                                                ; ES:DI = user byte if AL bit 4 clear
                                %endif
                                        ; 13/12/2023 - Retro DOS v5.0 IBMBIO.COM/IO.SYS
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:1CDAh)

                                        ; ........

00002133 B80041                         mov     ax, 4100h       ; wait for any external event (al=0)
00002136 B304                           mov     bl, 4           ; wait for 4 clock ticks
00002138 F9                             stc                     ; Assume we will fail
00002139 CD15                           int     15h             ; SYSTEM - WAIT ON EXTERNAL EVENT (CONVERTIBLE)
                                                                ; AL = condition type, BH = condition compare or mask value
                                                                ; BL = timeout value times 55 milliseconds, 00h means no timeout
                                                                ; DX = I/O port address if AL bit 4 set
                                        ; ........

0000213B 5F                             pop     di ; ?
0000213C 721B                           jc      short configdone ; 21/12/2022

0000213E C606[7900]01                   mov     byte [fhavek09], 1
                                                                ; remember we have a k09 type
00002143 1E                             push    ds
00002144 31C0                           xor     ax, ax
00002146 8ED8                           mov     ds, ax

00002148 893EB001                       mov     [6Ch*4], di
                                        ;mov    ds:1B0h, di     ; [6Ch*4]
                                                                ; new int 6Ch handler
                                        ;mov    word ptr ds:1B2h, cs ; [6Ch*4+2]
0000214C 8C0EB201                       mov     word [6Ch*4+2], cs
00002150 1F                             pop     ds
                                        ; 20/12/2022
                                        ; ds = cs = es
                                        ;mov    si, int6c
                                        ;mov    cx, endk09-int6c ; 459
                                        ;;mov   cx, 459         ; endk09 - int6c
                                                                ; size of k09 routine
                                        ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                        ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:2315h
00002151 BE[3E18]                       mov     si, int_6Ch
00002154 B9BC01                         mov     cx, endk09-int_6Ch ; 461 in PCDOS 7.1 IBMBIO.COM
00002157 F3A4                           rep movsb               ;
                                                                ; set up config       stuff for sysinit
                                ; ----------------------------------------------------------------------------
                                ; Set up config stuff for SYSINIT

                                ; 17/10/2022
                                ;SETDRIVE equ SetDrive - DOSBIOSEG_2C7h ; (4D7h for MSDOS 5.0 IO.SYS)
                                ;GETBP equ GetBp - DOSBIOSEG_2C7h ; (606h for MSDOS 5.0 IO.SYS)
                                ; 09/12/2022
                                SETDRIVE equ SetDrive
                                GETBP equ GetBp

                                        ; 17/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                configdone:
                                        ; 19/04/2024
                                        ;push   cs              ; di is final ending address of      msbio.
                                        ;pop    ds

0000215A 83C70F                         add     di, 15          ; round (up) to paragraph
                                        ; 10/12/2022
                                        ;shr    di, 1
                                        ;shr    di, 1
                                        ;shr    di, 1
                                        ;shr    di, 1
0000215C B104                           mov     cl, 4
0000215E D3EF                           shr     di, cl
                                        ; 10/12/2022
00002160 83C770                         add     di, 70h         ; KERNEL_SEGMENT (in fact: IO.SYS loading segment)
                                        ; 19/10/2022 - Temporary !
                                        ;db     81h, 0C7h, 70h, 0 ; add di, 0070h
00002163 893E[0300]                     mov     [DosDataSg], di         ; where the dos data segment will be

                                ; 11/12/2023 - Retro DOS v5.0 IBMBIO.COM/IO.SYS
                                ; ----------------------------------------------------------------------------
                                ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2332h)
                                ; ---------------------------------------
                                ; ("IBMDOS.COM" kernel file reading code here, below...)

00002167 A1[FA19]                       mov     ax, [drvfat]    ; get drive and       fat id
                                        ; 22/12/2022
                                        ; Note: SETDRIVES uses AL (drive number) only
0000216A BD[A405]                       mov     bp, SETDRIVE
                                        ;mov    bp, 5AEh ; 11/12/2023 (PCDOS 7.1 IBMBIO.COM)
                                        ;;mov   bp, 4D7h         ; set_drive (in       dosbios code segment)
                                                                ; at 2C7h:4D7h = 70h:2A47h
0000216D 0E                             push    cs              ; simulate far call
0000216E E800F9                         call    call_bios_code ; get bds for drive
```

```
                                                ; 06/04/2024
                                                ; es:di = BDS address
00002171 BD[D606]                       mov     bp, GETBP       ; ensure valid bpb is present
                                        ;mov    bp, 6E4h ; 11/12/2023 (PCDOS 7.1 IBMBIO.COM)
                                        ;;mov   bp, 606h        ; GetBp (2C7h:606h = 70h:2B76h)
00002174 0E                             push    cs
00002175 E8F9F8                         call    call_bios_code

                                ; resort to funky old segment definitions for now

                                        ; 22/12/2022
                                        ;push   es              ; copy bds to ds:di
                                        ;pop    ds

                                ; the following read of es:0000 was spurious anyway. Should look into it.
                                ;
                                ; hmmmmmm. j.k. took out a call to getfat right here a while
                                ;       back. Apparently it was what actually setup es: for the following
                                ; cas----

                                        ; 22/12/2022
                                        ;xor    di, di
                                        ;mov    al, [es:di]     ; get fat id byte
                                        ;;mov   byte ptr es:drvfat+1, al ; save fat byte
                                        ;mov    [es:drvfat+1], al
                                        ;mov    ax, [es:drvfat]

                                        ; 22/12/2022
                                        ; ds = cs
                                ;;;     mov     al, [drvfat]

                                ; cas -- why do a SECOND setdrive here???

                                        ; 22/12/2022
                                        ;push   es              ; save whatever's in es
                                        ;push   ds              ; copy bds to es:di
                                        ;pop    es
                                        ;push   cs              ; copy Bios_Data to ds
                                        ;pop    ds

                                ; 22/12/2022
                                ;;;     mov     bp, SETDRIVE
                                ;;;     ;mov    bp, 4D7h        ; SetDrive (2C7h:47Dh = 70h:2A47h)
                                ;;;     push    cs              ; simulate far call
                                ;;;     call    call_bios_code ; get correct bds for this drive

                                        ; 22/12/2022
                                        ;push   es              ; copy bds back to ds:di
                                        ;pop    ds
                                        ;pop    es              ; pop whatever was in es

                                ; Now we load in the MSDOS.SYS file

                                ; 22/12/2022
                                ; -----
                                ;       mov     bx, [di+6]      ; [di+BDS.BDS_BPB.BPB_BYTESPERSECTOR]
                                ;       mov     [cs:md_sectorsize], bx; used by get_fat_sector proc.
                                ;       mov     bl, [di+1Fh]    ; [di+BDS.fatsiz]
                                ;                               ; get size of fat on media
                                ;       ;mov    es:16DEh, bl
                                ;       mov     [es:fbigfat], bl
                                ;       mov     cl, [di+8]
                                ;       mov     ax, [di+17h] ; [di+BDS.BDS_BPB.BPB_HIDDENSECTORS]
                                ;       ;sub    es:16D8h, ax
                                ;       sub     [es:bios_l], ax         ; subtract hidden sectors since we
                                ;                               ; need a logical sector number that will
                                ;                               ; be used by getclus(diskrd procedure)
                                ;       mov     ax, [di+19h] ; [di+BDS.BDS_BPB.BPB_HIDDENSECTORS+2]
                                ;       ;sbb    es:16DAh, ax
                                ;       sbb     [es:bios_h], ax         ; subtract upper 16 bits of sector num
                                ; -----

                                        ; 11/12/2023 - Retro DOS v5.0 IO.SYS/IBMBIO.COM
                                ; -----; 22/12/2022
00002178 268B5D06                       mov     bx, [es:di+6]  ; [di+BDS.BDS_BPB.BPB_BYTESPERSECTOR]
0000217C 891E[081A]                     mov     [md_sectorsize], bx ; used by get_fat_sector proc.
                                        ; 11/12/2023 ; *
00002180 268A5D3B                       mov     bl, [es:di+3Bh]         ; [di+BDS.fatsiz] ; *
                                        ;mov    bl, [es:di+1Fh]         ; [di+BDS.fatsiz]
                                                                ; get size of fat on media
00002184 881E[061A]                     mov     [fbigfat], bl
00002188 268A4D08                       mov     cl, [es:di+8]
0000218C 268B4517                       mov     ax, [es:di+17h] ; [di+BDS.BDS_BPB.BPB_HIDDENSECTORS]
00002190 2906[FC19]                     sub     [First_Data_Sector], ax ; *
                                        ;sub    [bios_l], ax    ; subtract hidden sectors since we
                                                                ; need a logical sector number that will
                                                                ; be used by getclus(diskrd procedure)
00002194 268B4519                       mov     ax, [es:di+19h] ; [di+BDS.BDS_BPB.BPB_HIDDENSECTORS+2]
00002198 1906[FE19]                     sbb     [First_Data_Sector+2], ax ; *
                                        ;sbb    [bios_h], ax    ; subtract upper 16 bits of sector num
                                ; ------

0000219C 30ED                           xor     ch, ch  ; cx = sectors/cluster

                                ; the boot program has left the directory at 0:500h

                                        ; 11/12/2023 - - Retro DOS v5.0 IBMBIO.COM/IO.SYS
                                        ;push   di
0000219E 1E                             push    ds
                                        ;xor    di, di
                                        ;mov    ds, di
0000219F 31DB                           xor     bx, bx ; 0
000021A1 8EDB                           mov     ds, bx
000021A3 8B1E3A05                       mov     bx, [53Ah]
                                        ;mov    bx, ds:53Ah             ; (First cluster of the 2nd dir entry
                                                                ; of root directory in the buffer at 500h)
000021A7 1F                             pop     ds
000021A8 8B36[7E1A]                     mov     si, [firstcluster_hw] ; 11/12/2023
                                                                ; (32 bit cluster number for FAT32 fs)
                                        ;pop    ds
                                        ;pop    di

                                        ; 12/12/2023
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2397h)
                                        ; ...........
                                        ; ds = cs
000021AC A0[061A]                       mov     al, [fbigfat]
000021AF 50                             push    ax              ; (*) save fbigfat flags
000021B0 A0[FA19]                       mov     al, [drvfat]
000021B3 0A06[801A]                     or      al, [Boot_Drv]
000021B7 757B                           jnz     short boot_drv_fixed ; hard disk
                                boot_drv_removable:                     ; calculate cluster count and set fbig or fbigbig flag
000021B9 53                             push    bx              ; for removable drives
000021BA 51                             push    cx
```

```
                                        ; 28/12/2023
                                        ;push   dx ; (not necessary)
                                        ; 12/12/2023
000021BB 06                             push    es
000021BC 1F                             pop     ds
000021BD 8B450E                         mov     ax, [di+0Eh]    ; [di+BDS.totalsecs16]
000021C0 31D2                           xor     dx, dx
000021C2 09C0                           or      ax, ax
000021C4 7506                           jnz     short prep_totalsecs_ok
000021C6 8B451B                         mov     ax, [di+1Bh]    ; [di+BDS.totalsecs32]
000021C9 8B551D                         mov     dx, [di+1Dh]
                                prep_totalsecs_ok:
000021CC 2B4509                         sub     ax, [di+9]      ; [di+BDS.resectors]
000021CF 83DA00                         sbb     dx, 0
000021D2 50                             push    ax
000021D3 52                             push    dx
000021D4 8B5D11                         mov     bx, [di+11h]    ; [di+BDS.fatsecs16]
000021D7 31C0                           xor     ax, ax
000021D9 09DB                           or      bx, bx
000021DB 7506                           jnz     short prep_fatsecs_ok
000021DD 8B5D1F                         mov     bx, [di+1Fh]    ; [di+BDS.fatsecs32]
000021E0 8B4521                         mov     ax, [di+21h]
                                prep_fatsecs_ok:
000021E3 8A4D0B                         mov     cl, [di+0Bh]    ; ax:bx = 32 bit count of FAT sectors
                                                                ; [di+BDS.fats]
000021E6 30ED                           xor     ch, ch
000021E8 F7E1                           mul     cx
000021EA 91                             xchg    ax, cx
000021EB F7E3                           mul     bx
000021ED 01D1                           add     cx, dx
000021EF 89C3                           mov     bx, ax          ; cx:bx = total (2*) fat sectors
000021F1 5A                             pop     dx
000021F2 58                             pop     ax              ; dx:ax = totals sectors - reserved sectors
000021F3 29D8                           sub     ax, bx
000021F5 19CA                           sbb     dx, cx          ; dx:ax = data sectors (includes root dir sectors)
000021F7 8B5D0C                         mov     bx, [di+0Ch]    ; [di+BDS.direntries]
000021FA 83C30F                         add     bx, 15          ; 16 directory entries per sector
                                                                ; (round up sector count by adding 15)
000021FD B104                           mov     cl, 4           ; (rounded) dir entries / 16
000021FF D3EB                           shr     bx, cl
00002201 31C9                           xor     cx, cx
00002203 29D8                           sub     ax, bx
00002205 19CA                           sbb     dx, cx          ; dx:ax = data sectors (except root directory sectors)
                                                                ; (will be used for cluster count calculation)
00002207 8A4D08                         mov     cl, [di+8]      ; [di+BDS.secperclus]
                                        ; 12/12/2023
0000220A 0E                             push    cs
0000220B 1F                             pop     ds
0000220C 50                             push    ax              ; 32 bit division (data sectors / sector per cluster)
0000220D 89D0                           mov     ax, dx
0000220F 31D2                           xor     dx, dx
00002211 F7F1                           div     cx
00002213 89C3                           mov     bx, ax
00002215 58                             pop     ax
00002216 F7F1                           div     cx
00002218 09DB                           or      bx, bx          ; 32 bit cluster count if bx > 0
0000221A 7505                           jnz     short set_fbigbig_flag ; too big cluster number
0000221C 83F8F6                         cmp     ax, 0FFF6h
0000221F 7207                           jb      short set_fbig_flag
                                set_fbigbig_flag:
00002221 800E[061A]20                   or      byte [fbigfat], 20h ; FAT32 ; fbigbig
00002226 EB0A                           jmp     short set_fbig_flag_ok
                                ; ---------------------------------------------------------------------------

                                set_fbig_flag:
00002228 3DF60F                         cmp     ax, 0FF6h       ; 4096-10
                                                                ; is this 16-bit fat?
0000222B 7205                           jb      short set_fbig_flag_ok ; no, small fat
0000222D 800E[061A]40                   or      byte [fbigfat], 40h ; FAT16 ; fbig
                                set_fbig_flag_ok:
                                        ; 28/12/2023
                                        ;pop    dx
00002232 59                             pop     cx
00002233 5B                             pop     bx
                                boot_drv_fixed:
00002234 31FF                           xor     di, di

                                        ; cx = sectors/cluster
                                        ; si:bx = first cluster
                                        ; di = 0

                                        ; ...........
                                loadit:
00002236 B80405                         mov     ax, SYSINITSEG ; 46Dh
                                        ;mov    ax, 544h        ; 11/12/2023 - PCDOS 7.1 IBMBIO.COM
                                        ;;mov   ax, 46Dh        ; sysinit segment
00002239 8EC0                           mov     es, ax
0000223B 268E06[7302]                   mov     es, [es:CURRENTDOSLOCATION] ; 09/12/2022
                                        ;mov    es, [es:271h]

00002240 E86008                         call    getclus         ; read cluster at ES:DI (DI is updated)

                                ; ---------------------------------------------------------------------------

                                        ; 13/12/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2431h)

                                        ;test   byte [cs:fbigfat], 20h ; fbigbig ; FAT32 fs flag
00002243 F606[061A]20                   test    byte [fbigfat], 20h ; fbigbig ; FAT32 fs flag
                                        ;jz     short iseof
                                        ; 06/04/2024
00002248 750D                           jnz     short eofbigbig

                                ; ---------------------------------------------------------------------------
                                ; 06/04/2024
                                %if 1
                                        ; 13/12/2023
                                iseof:
                                        ;;test  byte [cs:fbigfat], fbig
                                        ;test   byte [cs:fbigfat], 40h ; fbig
                                        ; 12/12/2023
                                        ; ds = cs
0000224A F606[061A]40                   test    byte [fbigfat], 40h ; fbig
0000224F 750C                           jnz     short eofbig
00002251 81FBF70F                       cmp     bx, 0FF7h
00002255 EB09                           jmp     short iseofx
                                %endif
                                ; ---------------------------------------------------------------------------

                                eofbigbig:  ; si:bx = 32 bit cluster number
```

```
00002257 81FEFF0F                       cmp     si, 0FFFh
0000225B 7503                           jnz     short iseofx
                                        ;cmp    bx, 0FFF7h
                                        ;jmp    short iseofx
                                        ; 06/04/2024
                                        ;jmp    short eofbig

                                ; ---------------------------------------------------------------------------
                                ; 06/04/2024
                                %if 0
                                        ; 13/12/2023
                                iseof:
                                        ;;test  byte [cs:fbigfat], fbig
                                        ;test   byte [cs:fbigfat], 40h ; fbig
                                        ; 12/12/2023
                                        ; ds = cs
                                        test    byte [fbigfat], 40h ; fbig
                                        jnz     short eofbig
                                        cmp     bx, 0FF7h
                                        jmp     short iseofx
                                %endif
                                ; ---------------------------------------------------------------------------

                                eofbig:
0000225D 83FBF7                         cmp     bx, 0FFF7h
                                iseofx:
00002260 72D4                           jb      short loadit   ; keep loading until cluster = eof
                                ; ---------------------------------------------------------------------------

                                        ; 06/04/2024
                                        ;call   setdrvparms    ;

                                        ; 28/12/2023
00002262 58                             pop     ax              ; (*) restore fbigfat flags
                                                                ; (after loading DOS kernel)
                                        ; 06/04/2024
                                        ;mov    [cs:fbigfat], al
00002263 A2[061A]                       mov     [fbigfat], al

00002266 E80405                         call    setdrvparms    ; 06/04/2024

                                        ;;;jmp  far ptr 46Dh:267h ; jmp       SYSINIT_SEG:SYSINIT_START
                                        ;;jmp   far 46Dh:267h
                                        ; 12/12/2023
                                        ;jmp    far 544h:269h  ; (PCDOS 7.1 IBMBIO.COM)

00002269 EA[6902]0405                   jmp     SYSINITSEG:SYSINITSTART

                                ; ============== S U B   R O U T I N E =======================================

                                ; Following are subroutines to support resident device driver initialization
                                ;
                                ;M011 -- note:  deleted setup_bdsms and reset_bdsms here

                                ;   M035 -- begin changed section

                                ;****************************************************************************
                                ; module name: remap
                                ;
                                ; descriptive name: all the code for himem that could be separated from msbio
                                ;
                                ; function:  remap the bds chain to adjusted logical drive numbers (drive
                                ;        letters) if more than two diskette drives on the system.
                                ;
                                ;     scheme:  if more than 2 diskette drives, first map the bds structure
                                ;          as usual and then rescan the bds chain to adjust the drive
                                ;          letters. to do this, scan for disk drives and assign logical
                                ;          drive number starting from 2 and then rescan diskette drives
                                ;          and assign next to the last logical drive number of last disk
                                ;          drive to the 3rd and 4th diskette drives.

                                ; input:       none
                                ; exit:    drive letters have been remapped in bds chain
                                ; exit error:  none
                                ; called from: msinit
                                ;
                                ; notes:  this function  will be called only if more than 2 diskettes are
                                ;     found in the system
                                ;     this function assumes that there are no more than 26 drives assigned
                                ;       this is guaranteed by the code that creates bdss for partitions
                                ;     this function assumes that the first entries in the chain are
                                ;      floppy drives, and all the rest are hard drives
                                ;     will alter the boot drive if necessary to reflect remapping
                                ;
                                ;****************************************************************************

                                ; 17/10/2022
                                ; 02/10/2022
                                        ; 15/12/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2464h)
                                        ; (MSDOS 6.22 IO.SYS - BIOSDATA:1D9Eh)

                                remap:          ; proc near

                                        ; 15/12/2023
                                        ; ds = cs
                                        ;mov    di, [cs:start_bds] ; get first bds
0000226E 8B3E[1901]                     mov     di, [start_bds]

                                ; search for 1st fixed disk physical drive num

                                drive_loop:
00002272 807D0480                       cmp     byte [di+4], 80h ; [di+BDS.drivenum]
                                                                 ; first hard disk??
00002276 7409                           jz      short fdrv_found ; yes,       continue
00002278 8B3D                           mov     di, [di]         ; [di+BDS.link]
                                                                 ; get next bds,       assume segment
0000227A 83FFFF                         cmp     di, -1 ; 0FFFFh         ; last bds?
0000227D 75F3                           jnz     short drive_loop ; loop       if not
0000227F EB49                           jmp     short rmap_exit          ; yes, no hard drive on      system

                                ;---------------------------------------------------------------------------
                                ;first disk drive bds, now change the logical drive num to 2 and the subsequent
                                ;logical drive nums to 3, 4, 5 etc.
                                ;---------------------------------------------------------------------------

                                fdrv_found:
00002281 B002                           mov     al, 2           ; start with logical drv num=2
                                fdrv_loop:
00002283 884505                         mov     [di+5], al      ; [di+BDS.drivelet]
00002286 8B3D                           mov     di, [di]        ; [di+BDS.link]
                                                                ; ds:di--> next       bds
                                        ;inc    al              ; set num for next drive
                                        ; 18/12/2022
```

```
8311 00002288 40                            inc     ax
8312 00002289 83FFFF                        cmp     di, 0FFFFh      ; last hard drive ?
8313 0000228C 75F5                          jnz     short fdrv_loop         ; no - assign more disk drives
8314
8315                                ;---------------------------------------------------------------------------
8316                                ; now, rescan and find bds of 3rd floppy drive and assign next drive letter
8317                                ; in al to 3rd. if the current drive letter is past z, then do not allocate
8318                                ; any more.
8319                                ;---------------------------------------------------------------------------
8320
8321                                            ;mov    di, [cs:start_bds] ; [start_bds]
8322                                            ; 15/12/2023
8323 0000228E 8B3E[1901]                    mov     di, [start_bds]         ; get first bds
8324 00002292 8B3D                          mov     di, [di]        ; [di+BDS.link]
8325                                                                ; ds:di-->bds2
8326                                            ;mov    ah, [cs:dsktnum] ; get number of floppies to remap
8327 00002294 8A26[2501]                    mov     ah, [dsktnum]
8328 00002298 80EC02                        sub     ah, 2           ; adjust for a:       & b:
8329                                remap_loop1:
8330 0000229B 8B3D                          mov     di, [di]        ; [di+BDS.link]
8331                                                                ; set new num to next floppy
8332 0000229D 884505                        mov     [di+5], al      ; [di+BDS.drivelet]
8333 000022A0 FEC0                          inc     al              ; new number for next floppy
8334 000022A2 FECC                          dec     ah              ; count down extra floppies
8335 000022A4 75F5                          jnz     short remap_loop1
8336
8337                                ; now we've got to adjust the boot drive if we reassigned it
8338
8339                                            ; 15/12/2023
8340                                            ;mov    al, [cs:drvfat]
8341 000022A6 A0[FA19]                      mov     al, [drvfat]
8342 000022A9 3C02                          cmp     al, 2           ; is it a: or b: ?
8343 000022AB 721D                          jb      short rmap_exit
8344                                            ;sub    al, [cs:dsktnum]
8345 000022AD 2A06[2501]                    sub     al, [dsktnum]  ; is it one of the other floppies?
8346 000022B1 7204                          jb      short remap_boot_flop ;       brif so
8347
8348                                ; we've got to remap the boot hard drive
8349                                ; subtract the number of EXTRA floppies from it
8350
8351 000022B3 0402                                  add     al, 2           ; bootdrv -= (dsktnum-2)
8352 000022B5 EB04                                  jmp     short remap_change_boot_drv
8353                                ; ---------------------------------------------------------------------------
8354
8355                                ; we've got to remap the boot floppy.
8356                                ; add the number of hard drive partitions to it
8357
8358                                remap_boot_flop:
8359                                            ;add    al, [cs:drvmax]         ; bootdrv += (drvmax-dsktnum)
8360                                            ; 15/12/2023
8361 000022B7 0206[7500]                    add     al, [drvmax]
8362                                remap_change_boot_drv:
8363                                            ;mov    [cs:drvfat], al ; alter msdos.sys load drive
8364 000022BB A2[FA19]                      mov     [drvfat], al
8365 000022BE FEC0                          inc     al
8366 000022C0 1E                            push    ds
8367 000022C1 BF0405                        mov     di, SYSINITSEG ; 46Dh
8368                                            ;mov    di, 544h        ; PCDOS 7.1 IBMBIO.COM
8369                                            ;;mov   di, 46Dh        ; SYSINIT segment
8370 000022C4 8EDF                          mov     ds, di
8371 000022C6 A2[9802]                      mov     [DEFAULTDRIVE], al
8372                                            ;mov    ds:296h, al     ; [SYSINIT+DEFAULT_DRIVE]
8373                                                                    ; pass it to sysinit as       well
8374 000022C9 1F                            pop     ds ; ds = cs
8375                                rmap_exit:
8376 000022CA C3                            retn
8377
8378                                ; =============== S U B   R O U T I N E =======================================
8379
8380                                ; 17/10/2022
8381                                ; 02/10/2022 - Retro DOS v4.0 (MSDOS 5.0 -actual-)
8382                                ; 28/12/2018 - Retro DOS v4.0 (MSDOS 6.21 -draft-)
8383                                ; 02/06/2018 - Retro DOS v3.0 (MSDOS 3.3)
8384                                ; 19/03/2018 - Retro DOS v2.0 (MSDOS 2.11)
8385                                ;************************************************
8386                                ; getboot - get the boot sector for a hard disk
8387                                ;
8388                                ; Reads the boot sector from a specified drive into
8389                                ; a buffer at the top of memory.
8390                                ;
8391                                ; dl = int13 drive number to read boot sector for
8392                                ;************************************************
8393
8394                                ; 17/10/2022
8395                                bootbias equ 200h
8396
8397                                getboot:    ; proc near
8398
8399                                            ; 15/12/2023 - Retro DOS v5.0
8400                                            ;       (Modified PCDOS 7.1) IBMBIO.COM/IO.SYS
8401                                            ; ds = cs
8402
8403                                            ; 08/04/2018
8404                                            ; Retro DOS v2.0 (IBMBIO.COM, IBMDOS 2.1)
8405                                            ; 28/03/2018 - MSDOS 6.0 - MSINIT.ASM, 1991
8406                                            ; 02/10/2022 - Retro DOS v4.0
8407                                            ;             (disassembled IO.SYS code of MSDOS 5.0)
8408
8409                                            ;mov    ax, [cs:init_bootseg] ; 17/10/2022
8410                                            ; 15/12/2023
8411 000022CB A1[041A]                      mov     ax, [init_bootseg]
8412 000022CE 8EC0                          mov     es, ax
8413
8414                                            ; 17/10/2022
8415 000022D0 BB0002                        mov     bx, bootbias ; 200h
8416                                            ;mov    bx, 200h        ; bootbias
8417                                                                    ; load BX, ES:BX is where sector goes
8418 000022D3 B80102                        mov     ax, 201h
8419 000022D6 30F6                          xor     dh, dh
8420 000022D8 B90100                        mov     cx, 1
8421 000022DB CD13                          int     13h             ; DISK - READ SECTORS INTO MEMORY
8422                                                                ; AL = number of sectors to read, CH = track, CL = sector
8423                                                                ; DH = head, DL       = drive, ES:BX -> buffer to fill
8424                                                                ; Return: CF set on error, AH =      status, AL = number of sectors read
8425 000022DD 7209                          jc      short erret
8426                                            ; 17/10/2022
8427 000022DF 26813EFE0355AA                cmp     word [es:bootbias+1FEh], 0AA55h
8428                                            ;cmp    word ptr es:3FEh, 0AA55h ; [es:bootbias+1FEh]
8429                                                                ; Dave Litton magic word?
8430 000022E6 7401                          jz      short norm_ret ; yes
8431                                erret:
8432 000022E8 F9                            stc
8433                                norm_ret:
8434 000022E9 C3                            retn
```

```
                                ; ============== S U B   R O U T I N E =======================================

                                ; 28/12/2018 - Retro DOS v4.0

                                ;//////////////////////////////////////////////////////////////////////////////
                                ;
                                ;   sethard - generate bpb for a variable sized hard file. ibm has a
                                ;   partitioned hard file; we must read physical sector 0 to determine where
                                ;   our own logical sectors start. we also read in our boot sector to
                                ;   determine version number
                                ;
                                ;   inputs: dl is rom drive number (80...)
                                ;           bh is partition number (0....)
                                ;           ds:di points to bds
                                ;   outputs:       carry clear -> bpb is filled in
                                ;           carry set   -> bpb is left uninitialized due to error
                                ;   trashes (at least) si, cx
                                ;   MUST PRESERVE ES:!!!!
                                ;//////////////////////////////////////////////////////////////////////////////
                                ;
                                        ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:24E9h)

                                sethard:    ; proc near
                                        ; 12/08/2023
                                        ; ds = cs = BIOSDATA
000022EA 57                             push    di
000022EB 53                             push    bx
                                        ;push  ds  ; ds = cs = BIOSDATA ; 12/08/2023
000022EC 06                             push    es
000022ED 885D05                         mov     [di+5],bl       ; [di+BDS.drivelet]
000022F0 885504                         mov     [di+4],dl       ; [di+BDS.drivenum]
                                        ; 16/12/2023
000022F3 804D3F01                       or      byte [di+3Fh], 1 ; PCDOS 7.1
                                        ;or     byte [di+23h], 1 ; [di+BDS.flags]
                                                                 ; fnon_removable
000022F7 C6453E05                       mov     byte [di+3Eh], 5 ; PCDOS 7.1
                                        ;mov    byte [di+22h], 5 ; [di+BDS.formfactor]
                                                                 ; ffHardFile
000022FB C606[061A]00                   mov     byte [fbigfat], 0 ; assume 12 bit FAT
00002300 88FE                           mov     dh, bh          ; partition number
00002302 52                             push    dx
00002303 B408                           mov     ah, 8
00002305 CD13                           int     13h             ; DISK - DISK -       GET CURRENT DRIVE PARAMETERS (XT,AT,XT286,CONV,PS)
                                                                 ; DL = drive number
                                                                 ; Return: CF set on error, AH =      status code, BL       = drive type
                                                                 ; DL = number of consecutive drives
                                                                 ; DH = maximum value for head number, ES:DI -> drive parameter
                                        ;inc    dh
                                        ; 16/12/2023 - Retro DOS v5.0
00002307 88F2                           mov     dl, dh
00002309 B600                           mov     dh, 0
0000230B 42                             inc     dx
                                        ;mov    [di+15h], dh   ; [di+BDS.heads] ; get number of heads
0000230C 895515                         mov     [di+15h], dx
0000230F 5A                             pop     dx
00002310 7253                           jc      short setret   ; error if no hard disk
                                        ; 16/12/2023
                                        ;jc     short setret_j

                                        ; 17/04/2024
                                        ;and    cx, 3Fh
                                        ;mov    [di+13h],cx
00002312 80E13F                         and     cl, 3Fh
00002315 884D13                         mov     [di+13h], cl   ; [di+BDS.secpertrack]

00002318 52                             push    dx              ; save partition number
00002319 E8AFFF                         call    getboot
0000231C 5A                             pop     dx              ; restore partition number
0000231D 7246                           jc      short setret
                                        ; 16/12/2023
                                        ;jnc    short chk_act_part
                                ;setret_j:
                                        ;jmp    setret

                                        ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                chk_act_part:
0000231F 31DB                           xor     bx, bx ; 0
                                        ;;mov   [cs:ep_start_sector], bx
                                        ;;mov   [cs:ep_start_sector+2], bx
                                        ;mov    [cs:ep_hidden_secs], bx
                                        ;mov    [cs:ep_hidden_secs+2], bx
                                        ; 16/12/2023
                                        ; ds = cs
                                        ; 20/12/2023
                                        ;mov    [ep_start_sector], bx
                                        ;mov    [ep_start_sector+2], bx
00002321 891E[8723]                     mov     [ep_hidden_secs], bx
00002325 891E[8923]                     mov     [ep_hidden_secs+2], bx

00002329 BBC203                         mov     bx, 3C2h        ; 1C2h+bootbias

                                ; The first 'active' partition is 00, the second is 01....
                                ;   then the remainder of the 'primary' but non-active partitions

                                act_part:
0000232C 26F647FC80                     test    byte [es:bx-4], 80h ; is the partition active?
00002331 740B                           jz      short no_act    ; no
                                ; 16/12/2023
                                %if 0
                                        ; 16/12/2023
                                        ; reject if partitiontype != 1, 4, 6, 0Bh, 0Ch, 0Eh
                                        cmp     byte [es:bx], 1 ; FAT12
                                        jz      short got_good_act
                                        cmp     byte [es:bx], 4         ; FAT16 CHS (<= 32MB)
                                        jz      short got_good_act

                                        ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                        cmp     byte [es:bx], 0Bh ; FAT32 CHS
                                        jz      short got_good_act
                                        cmp     byte [es:bx], 0Ch ; FAT32 LBA
                                        jz      short got_good_act
                                        cmp     byte [es:bx], 0Eh ; FAT16 LBA
                                        jz      short got_good_act

                                        cmp     byte [es:bx], 6         ; FAT16 BIG CHS (> 32MB)
                                        jnz     short no_act
                                ;%else
                                        ; 16/12/2023
                                        mov     al, [es:bx]     ; partition type

                                        ; reject if partitiontype != 1, 4, 6, 0Bh, 0Ch, 0Eh
                                        cmp     al, 1           ; FAT12
                                        je      short got_good_act
```

```
                                        cmp     al, 4           ; FAT16 CHS (<= 32MB)
                                        je      short got_good_act

                                        ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                        cmp     al, 0Bh         ; FAT32 CHS
                                        je      short got_good_act
                                        cmp     al, 0Ch         ; FAT32 LBA
                                        je      short got_good_act
                                        cmp     al, 0Eh         ; FAT16 LBA
                                        je      short got_good_act

                                        cmp     al, 6           ; FAT16 BIG CHS (> 32MB)
                                        jne     short no_act
                                %endif
                                        ; 16/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
                                        ; check if it is a primary dos partition

00002333 E83300                         call    chk_partition_type
00002336 7506                           jne     short no_act

                                got_good_act:                           ; 11/08/2023
00002338 08F6                           or      dh, dh          ; is this our target partition #?
                                                                ; (0 = first primary dos or active partition)
0000233A 744F                           jz      short set2      ; WE GOT THE ONE WANTED!!
0000233C FECE                           dec     dh              ; count down
                                no_act:
0000233E 83C310                         add     bx, 16
00002341 81FB0204                       cmp     bx, 402h        ; 202h+bootbias
                                                                ; last entry done?
00002345 75E5                           jnz     short act_part ; no, process next entry

00002347 BBC203                         mov     bx, 3C2h        ; 1C2h+bootbias
                                                                ; restore original value of bx

                                ; Now scan the non-active partitions

                                get_primary:
0000234A 26F647FC80                     test    byte [es:bx-4], 80h
0000234F 750B                           jnz     short not_prim ; we've already scanned
                                                                ; the ACTIVE ones
                                ; 16/12/2023
                                %if 0
                                        ; 16/12/2023
                                        ; see if partitiontype == 1, 4, 6, 0Bh, 0Ch, 0Eh
                                        cmp     byte [es:bx], 1         ; FAT12
                                        jz      short got_prim
                                        cmp     byte [es:bx], 4         ; FAT16 CHS (<= 32MB)
                                        jz      short got_prim

                                        ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                        cmp     byte [es:bx], 0Bh ; FAT32 CHS
                                        jz      short got_prim
                                        cmp     byte [es:bx], 0Ch ; FAT32 LBA
                                        jz      short got_prim
                                        cmp     byte [es:bx], 0Eh ; FAT16 LBA
                                        jz      short got_prim

                                        cmp     byte [es:bx], 6         ; FAT16 BIG CHS (> 32MB)
                                        jnz     short not_prim
                                ;%else
                                        ; 16/12/2023
                                        mov     al, [es:bx]     ; partition type

                                        ; see if partitiontype == 1, 4, 6, 0Bh, 0Ch, 0Eh
                                        cmp     al, 1           ; FAT12
                                        je      short got_prim
                                        cmp     al, 4           ; FAT16 CHS (<= 32MB)
                                        je      short got_prim

                                        ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                        cmp     al, 0Bh         ; FAT32 CHS
                                        je      short got_prim
                                        cmp     al, 0Ch         ; FAT32 LBA
                                        je      short got_prim
                                        cmp     al, 0Eh         ; FAT16 LBA
                                        je      short got_prim

                                        cmp     al, 6           ; FAT16 BIG CHS (> 32MB)
                                        jne     short not_prim
                                %endif
                                        ; 16/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
                                        ; check if it is a primary dos partition

00002351 E81500                         call    chk_partition_type
00002354 7506                           jne     short not_prim

                                got_prim:
00002356 08F6                           or      dh, dh          ; is this our target partition?
00002358 7431                           jz      short set2
0000235A FECE                           dec     dh
                                not_prim:
0000235C 83C310                         add     bx, 16
0000235F 81FB0204                       cmp     bx, 402h        ; 202h+bootbias
00002363 75E5                           jne     short get_primary ; loop till we've gone through table
                                setret:
00002365 F9                             stc                     ; error return
00002366 E9C703                         jmp     ret_hard_err

                                ; ---------------------------------------------------------------------------

                                        ; 16/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)

                                chk_partition_type:
                                        ; 16/12/2023
00002369 268A07                         mov     al, [es:bx]     ; partition type

                                        ; see if partitiontype == 1, 4, 6, 0Bh, 0Ch, 0Eh
0000236C 3C01                           cmp     al, 1           ; FAT12
0000236E 7412                           je      short chk_ptype_retn
00002370 3C04                           cmp     al, 4           ; FAT16 CHS (<= 32MB)
00002372 740E                           je      short chk_ptype_retn

                                        ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
00002374 3C0B                           cmp     al, 0Bh         ; FAT32 CHS
00002376 740A                           je      short chk_ptype_retn
00002378 3C0C                           cmp     al, 0Ch         ; FAT32 LBA
0000237A 7406                           je      short chk_ptype_retn
0000237C 3C0E                           cmp     al, 0Eh         ; FAT16 LBA
0000237E 7402                           je      short chk_ptype_retn

00002380 3C06                           cmp     al, 6           ; FAT16 BIG CHS (> 32MB)
                                chk_ptype_retn:
                                        ; zf = 1 -> primary DOS partition
                                        ; zf = 0 -> not a primary DOS partition
```

```
00002382 C3                             retn
                                ; ---------------------------------------------------------------------------
                                        ; 16/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:25B6h)
                                ep_start_sector:
00002383 00000000                       dd 0
00002387 00000000               ep_hidden_secs:    dd 0
                                ; ---------------------------------------------------------------------------
                                ;  until we get the real logical boot record and get the bpb,
                                ;  BDS_BPB.BPB_BIGTOTALSECTORS will be used instead of BDS_BPB.BPB_TOTALSECTORS
                                ;  for the convenience of the computation.
                                ;
                                ;  at the end of this procedure, if a bpb information is gotten from
                                ;  the valid boot record, then we are going to use those bpb information
                                ;  without change.
                                ;
                                ;  otherwise, if (hidden sectors + total sectors) <= a word, then we will move
                                ;  BDS_BPB.BPB_BIGTOTALSECTORS (low) to BDS_BPB.BPB_TOTALSECTORS and zero out
                                ;  BDS_BPB.BPB_BIGTOTALSECTORS entry to make it a conventional bpb format.
                                set2:
                                        ; 12/08/2023
                                        ; ds = cs = BIOSDATA segment (0070h)
0000238B 8816[071A]                     mov     [rom_drv_num], dl
                                        ;mov    [cs:rom_drv_num], dl
                                                ; save the rom bios drive number we are handling now.
0000238F 268B4704                       mov     ax, [es:bx+4]  ; hidden sectors (start      sector)
00002393 268B5706                       mov     dx, [es:bx+6]
                                ; decrement the sector count by 1 to make it zero based. exactly 64k
                                ; sectors should be allowed
00002397 83E801                         sub     ax, 1
0000239A 83DA00                         sbb     dx, 0
0000239D 26034708                       add     ax, [es:bx+8]  ; sectors in partition
000023A1 2613570A                       adc     dx, [es:bx+10]
000023A5 7305                           jnc     short okdrive
000023A7 800E[061A]80                   or      byte [fbigfat], 80h ; ftoobig
                                        ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
                                        ;;;
                                okdrive:
                                        ;add    ax, [cs:ep_hidden_secs]
                                        ;adc    dx, [cs:ep_hidden_secs+2]
                                        ; ds = cs
000023AC 0306[8723]                     add     ax, [ep_hidden_secs]
000023B0 1316[8923]                     adc     dx, [ep_hidden_secs+2]
000023B4 7305                           jnc     short okdrive_1
000023B6 800E[061A]80                   or      byte [fbigfat], 80h ; ftoobig
                                okdrive_1:
000023BB 26803F0C                       cmp     byte [es:bx], 0Ch ; FAT32 LBA partition ID
000023BF 7418                           je      short set_lba_flag
000023C1 26803F0E                       cmp     byte [es:bx], 0Eh ; FAT16 LBA partition ID
000023C5 7412                           je      short set_lba_flag
000023C7 3B5513                         cmp     dx, [di+13h]   ; if dx > [di+BDS.secpertrack] then
000023CA 730D                           jnb     short set_lba_flag ; set LBA r/w flag
000023CC F77513                         div     word [di+13h]
000023CF 31D2                           xor     dx, dx
000023D1 F77515                         div     word [di+15h]
000023D4 3D0004                         cmp     ax, 400h        ; if ax (cylinder number) >= 1024
                                                                ;  set LBA r/w flag
000023D7 7204                           jb      short set3
                                set_lba_flag:
000023D9 804D4004                           or byte [di+40h], 4 ; fLBArw ; LBA r/w flag
                                        ;;;
                                ;okdrive:
                                        ; 16/12/2023
                                set3:
                                        ;mov    ax, [es:bx+4]
                                        ;mov    [di+17h], ax   ; [di+BDS.hiddensecs]
                                        ;                      ; BPB_HIDDENSECTORS = p->partitionbegin
                                        ;mov    ax, [es:bx+6]
                                        ;mov    [di+19h], ax   ; [di+BDS.hiddensecs+2]
                                        ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
                                        ;;;
000023DD 268B4704                       mov     ax, [es:bx+4]  ; start sector (LBA) of the partition
000023E1 268B5706                       mov     dx, [es:bx+6]
                                        ;add    ax, [cs:ep_hidden_secs]
                                        ;adc    dx, [cs:ep_hidden_secs+2]
                                        ; ds = cs
000023E5 0306[8723]                     add     ax, [ep_hidden_secs]
                                                        ; + hidden secs of the extd dos partion
000023E9 1316[8923]                     adc     dx, [ep_hidden_secs+2]
000023ED 894517                         mov     [di+17h], ax   ; [di+BDS.hiddensecs]
000023F0 895519                         mov     [di+19h], dx   ; [di+BDS.hiddensecs+2]
000023F3 31C0                           xor     ax, ax ; 0
000023F5 89457B                         mov     [di+7Bh], ax   ; [di+BDS.bdsm_hidden_trks]
000023F8 89450E                         mov     [di+0Eh], ax   ; [di+BDS.totalsec16]
                                        ;;;
000023FB 268B570A                       mov     dx, [es:bx+10] ; # of sectors (high)
000023FF 268B4708                       mov     ax, [es:bx+8]  ; # of sectors (low)
00002403 89551D                         mov     [di+1Dh], dx   ; [di+BDS.totalsecs32+2]
00002406 89451B                         mov     [di+1Bh], ax   ; [di+BDS.totalsecs32]
                                                                ; bpb->maxsec =       p->partitionlength
                                        ;cmp    dx, 0
                                        ;ja     short okdrive_1
                                        ; 16/12/2023
00002409 09D2                           or      dx, dx
0000240B 7505                           jnz     short set3_read
0000240D 83F840                         cmp     ax, 64         ; if (p->partitionlength < 64)
                                        ;jb     short setret   ; return -1;
00002410 7264                           jb      short set3_err
                                ;okdrive_1:
                                        ; 16/12/2023
                                set3_read:
00002412 8B5519                         mov     dx, [di+19h]   ; [di+BDS.hiddensecs+2]
00002415 8B4517                         mov     ax, [di+17h]   ; [di+BDS.hiddensecs]
00002418 31DB                           xor     bx, bx         ; boot sector number - for mini      disk
                                                               ; usually equal       to the # of sec/trk.
0000241A 8A5D13                         mov     bl, [di+13h]   ; [di+BDS.secpertrack]
0000241D 50                             push    ax
0000241E 89D0                           mov     ax, dx
00002420 31D2                           xor     dx, dx
00002422 F7F3                           div     bx             ; (sectors)dx:ax / (BDS.secpertrack)bx =
                                                               ; (track)temp_h:ax + (sector)dx
                                ; 16/12/2023
                                %if 0
                                        ; 17/10/2022
```

```
                                ;mov    [cs:temp_h], ax
                                ; 12/08/2023 (ds=cs)
                                mov     [temp_h], ax
                                pop     ax
                                div     bx
                                mov     cl, dl
                                inc     cl
                                xor     bx, bx
                                mov     bl, [di+15h]   ; [di+BDS.heads]
                                push    ax
                                xor     dx, dx
                                ;mov    ax, [cs:temp_h]
                                mov     ax, [temp_h] ; 12/08/2023
                                div     bx
                                ;mov    [cs:temp_h], ax
                                mov     [temp_h], ax ; 12/08/2023
                                pop     ax
                                div     bx              ; dl is head, ax is cylinder
                                ; 12/08/2023 (ds=cs)
                                cmp     word [temp_h], 0
                                ;cmp    word [cs:temp_h], 0
                                ja      short setret_brdg ; exceeds the      limit of int 13h
                                cmp     ax, 1024
                                ja      short setret_brdg ; exceeds the      limit of int 13h
                                        ; Retro DOS v3.2 note by Erdogan Tan - 28/07/2019
                                        ; **MSDOS code accepts if ax = 1024 but it is nonsense here
                                        ; ('ja' must be 'jnb')
                        okdrive_2:
                                ; 28/07/2019
                        ; dl is head.
                        ; ax is cylinder
                        ; cl is sector number (assume less than 2**6 = 64 for int 13h)

                        ;*** for mini disks ***

                                cmp     word [di+47h], 1 ; [di+BDS.bdsm_ismini]
                                                         ; check for mini disk
                                jnz     short oknotmini       ; not mini disk.
                                add     ax, [di+49h]   ; [di+BDS.bdsm_hidden_trks]
                                                         ; set the physical track number
                        oknotmini:
                        %endif
                                ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
                                ;;;
                                ;mov    [cs:saved_word], ax
00002424 A3[9E04]               mov     [saved_word], ax
00002427 58                     pop     ax
00002428 F7F3                   div     bx
0000242A 88D1                   mov     cl, dl
0000242C FEC1                   inc     cl
0000242E 8B5D15                 mov     bx, [di+15h]   ; [di+BDS.heads]
00002431 50                     push    ax
00002432 31D2                   xor     dx, dx
                                ;mov    ax, [cs:saved_word]
00002434 A1[9E04]               mov     ax, [saved_word]
00002437 F7F3                   div     bx
                                ;mov    [cs:saved_word], ax
00002439 A3[9E04]               mov     [saved_word], ax ; not necessary !? (ax must be 0)
0000243C 58                     pop     ax
0000243D F7F3                   div     bx              ; dl is head, ax is cylinder
                                ; 16/12/2023
0000243F 0E                     push    cs
00002440 07                     pop     es ; (*)
00002441 BB[5201]               mov     bx, disksector ; (**)
                                ;
00002444 F6454004               test    byte [di+40h], 4 ; fLBArw ; LBA read/write flag
00002448 742F                   jz      short set3_chs_read
                        set3_lba_read:

                        ; 16/12/2023
                        %if 0
                                ;push   cs
                                ;pop    es ; (*)
                                ;mov    bx, disksector ; (**)

                                ;push   ds
                                ;push   si
                                xor     ax, ax  ; 0
                                push    ax
                                push    ax
                                mov     ax, [di+19h]   ; [di+BDS.hiddensectors+2]
                                push    ax
                                mov     ax, [di+17h]   ; [di+BDS.hiddensectors]
                                push    ax
                                push    es              ; buffer address
                                push    bx
                                mov     ax, 1          ; sector (read) count
                                push    ax
                                ;mov    ax, 16         ; DAP size
                                mov     al, 16
                                push    ax
                                mov     dl, [rom_drv_num] ; ds = cs
                                mov     ax, ss
                                mov     ds, ax ; ds = ss
                                mov     si, sp
                                ;mov    dl, [cs:rom_drv_num]
                                mov     ah, 42h
                                int     13h             ; DISK - IBM/MS Extension
                                                        ; EXTENDED READ (DL - drive, DS:SI - disk address packet)
                                ;pop    si
                                ;pop    ds
                                jnc     short set3_lba_read_ok
                                add     sp, 16
                                ;pop    si
                                ;pop    ds
                        set3_err:
                                ;jmp    setret
                                jmp     ret_hard_err

                        set3_lba_read_ok
                                add     sp, 16
                                ;pop    si
                                ;pop    ds
                                jmp     short set3_read_ok
                        %else
                                ; 16/12/2023
                                ;push   si ; * ; (not necessary)
                                ;mov    si, empty_dap_buff ; dap_buffer
0000244A BE[571B]               mov     si, dap_buffer ; empty_dap_buff
0000244D 56                     push    si
0000244E 87FE                   xchg    si, di
                                ; si = BDS
                                ; di = DAP buffer
00002450 B81000                 mov     ax, 16
```

```
8931 00002453 AB                            stosw           ; DAP size
8932 00002454 B001                          mov     al, 1
8933 00002456 AB                            stosw           ; sector (read) count
8934                                        ; buffer address
8935 00002457 89D8                          mov     ax, bx  ; offset disksector
8936 00002459 AB                            stosw
8937 0000245A 8CC0                          mov     ax, es  ; es=ds=cs = BIOSDATA segment
8938 0000245C AB                            stosw
8939                                        ; sector address (bits 0 to 31)
8940 0000245D 8B4417                        mov     ax, [si+17h] ; [di+BDS.hiddensectors]
8941 00002460 AB                            stosw
8942 00002461 8B4419                        mov     ax, [si+19h] ; [di+BDS.hiddensectors+2]
8943 00002464 AB                            stosw
8944                                        ; sector address bits 32 to 63 (0)
8945 00002465 31C0                          xor     ax, ax ; 0
8946 00002467 AB                            stosw
8947 00002468 AB                            stosw
8948                                        ;xchg   di, si
8949 00002469 89F7                          mov     di, si
8950                                        ; di = BDS
8951 0000246B 5E                            pop     si ; DAP buffer address
8952
8953 0000246C 8A16[071A]                    mov     dl, [rom_drv_num] ; ds = cs
8954 00002470 B442                          mov     ah, 42h
8955 00002472 CD13                          int     13h             ; DISK - IBM/MS Extension
8956                                                                ; EXTENDED READ (DL - drive, DS:SI - disk address packet)
8957                                        ;pop    si ; *
8958 00002474 7324                          jnc     short set3_read_ok
8959                                set3_err:
8960                                        ;jmp    setret
8961 00002476 E9B702                        jmp     ret_hard_err
8962                                %endif
8963
8964                                set3_chs_read:
8965 00002479 837D7901                      cmp     word [di+79h], 1 ; [di+BDS.bdsm_ismini] ; check for mini disk
8966 0000247D 7503                          jnz     short oknotmini
8967 0000247F 03457B                        add     ax, [di+7Bh]    ; [di+BDS.bdsm_hidden_trks]
8968                                        ;;;
8969
8970                                oknotmini:
8971                                ;*** end of added logic for mini disk
8972
8973 00002482 D0CC                          ror     ah, 1           ; move high two bits of cyl to high
8974 00002484 D0CC                          ror     ah, 1           ; two bits of upper byte
8975 00002486 80E4C0                        and     ah, 0C0h        ; turn off remainder of bits
8976 00002489 08E1                          or      cl, ah          ; move two bits to correct spot
8977 0000248B 88C5                          mov     ch, al          ; ch iscylinder (low 8 bits)
8978                                                                ; cl is sector + 2 high bits of cylinder
8979 0000248D 88D6                          mov     dh, dl          ; dh is head
8980
8981                                        ; 12/08/2023 (ds=cs)
8982 0000248F 8A16[071A]                    mov     dl, [rom_drv_num]
8983                                        ;mov    dl, [cs:rom_drv_num] ; dl is drive number
8984
8985                                ; cl is sector + 2 high bits of cylinder
8986                                ; ch is low 8 bits of cylinder
8987                                ; dh is head
8988                                ; dl is drive
8989
8990                                ; for convenience, we are going to read the logical boot sector
8991                                ; into cs:disksector area.
8992
8993                                ; read in boot sector using bios disk interrupt. the buffer where it
8994                                ; is to be read in is cs:disksector.
8995
8996                                        ; 16/12/2023
8997                                        ; es=ds=cs = BIOSDATA segment
8998                                        ; bx = disksector ; (**)
8999
9000                                        ;push   cs
9001                                        ;pop    es ; (*)
9002
9003                                        ;mov    bx, disksector ; for convenience,
9004                                                                ; we are going to read the logical boot sector
9005                                                                ; into cs:disksector area.
9006 00002493 B80102                        mov     ax, 201h
9007 00002496 CD13                          int     13h             ; DISK - READ SECTORS INTO MEMORY
9008                                                                ; AL = number of sectors to read, CH = track, CL = sector
9009                                                                ; DH = head, DL       = drive, ES:BX -> buffer to fill
9010                                                                ; Return: CF set on error, AH =     status, AL = number of sectors read
9011                                        ; 16/12/2023
9012 00002498 72DC                          jc      short set3_err
9013
9014                                ; cs:disksec contains the boot sector. in theory, (ha ha) the bpb in this thing
9015                                ; is correct. we can, therefore, suck out all the relevant statistics on the
9016                                ; media if we recognize the version number.
9017
9018                                set3_read_ok:
9019                                        ; 11/08/2023
9020                                        ;mov    bx, disksector ; BIOSDATA:014Eh ; MSDOS 6.21 ; 11/08/2023
9021                                                                ; BIOSDATA:0152h ; PCDOS 7.1 IBMBIO.COM
9022                                        ; 18/12/2023
9023                                        ;push   bx ; +
9024                                        ;push   ax ; (not necessary)
9025
9026                                        ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
9027                                        ;;;
9028 0000249A 81BFFE0155AA                  cmp     word [bx+1FEh], 0AA55h
9029 000024A0 7541                          jne     short invalid_boot_record
9030                                        ; 16/12/2023
9031                                        ; 12/08/2023
9032                                        ; ds = cs = BIOSDATA segment ('disksector:' is in BIOSDATA)
9033 000024A2 803FE9                        cmp     byte [bx], 0E9h         ; is it a near jump?
9034 000024A5 740B                          je      short check_1_ok ; yes
9035 000024A7 803FEB                        cmp     byte [bx], 0EBh         ; is it a short jump?
9036 000024AA 7537                          jne     short invalid_boot_record ; no
9037 000024AC 807F0290                      cmp     byte [bx+2], 90h ; yes, is the next one a nop?
9038 000024B0 7531                          jne     short invalid_boot_record ; no, invalid bs ; 11/08/2023
9039                                check_1_ok:
9040 000024B2 837F1600                      cmp      word [bx+16h], 0 ; [bx+BPB_FATSz16]
9041                                        ;jz     short check_1  ; 16 bit FAT size is 0 if it is FAT32 bs
9042                                        ; 16/12/2023
9043 000024B6 740E                          jz      short check_2  ; FAT32 bs
9044
9045                                        ; FAT16 or FAT12 bs
9046
9047                                        ;push   ds
9048                                        ;push   si  ; (not necessary)
9049 000024B8 57                            push    di
9050                                        ; es=ds=cs = BIOSDATA segment
9051                                        ;push   es
9052                                        ;pop    ds
9053
9054                                        ;mov    cx, 28
```

```
9055 000024B9 B90E00                            mov     cx, 14 ; *
9056 000024BC 8D7724                            lea     si, [bx+24h]   ; move offset 36 to 63
9057                                                                   ;       to offset 64 (28 bytes)
9058 000024BF 8D7F40                            lea     di, [bx+40h]   ; boot sector offset 64
9059 000024C2 FC                                cld     ; (not necessary, 'std' is not used before here)
9060                                            ;rep movsb
9061 000024C3 F3A5                              rep movsw ; *
9062 000024C5 5F                                pop     di
9063                                            ;pop    si
9064                                            ;pop    ds
9065                                            ;;;
9066                                    ; 16/12/2023
9067                                    %if 0
9068                                    ;check_1:
9069                                            ; 12/08/2023
9070                                            ; ds = cs = BIOSDATA segment ('disksector:' is in BIOSDATA)
9071                                            cmp     byte [bx], 0E9h
9072                                            ;cmp    byte [cs:bx], 0E9h ; is it a near jump?
9073                                            je      short check_1_ok ; yes
9074                                            cmp     byte [bx], 0EBh
9075                                            ;cmp    byte [cs:bx], 0EBh ; is it a short jump?
9076                                            jne     short invalid_boot_record ; no
9077                                            cmp     byte [bx+2], 90h
9078                                            ;cmp    byte [cs:bx+2], 90h ; yes, is the next one a nop?
9079                                            jne     short invalid_boot_record ; no, invalid bs ; 11/08/2023
9080                                    check_1_ok:
9081                                    %endif
9082
9083                                    ; 18/12/2023
9084                                    %if 0
9085                                            ; 14/08/2023
9086                                    check_2:
9087                                            mov     bx, disksector+11 ; disksector+EXT_BOOT.BPB
9088                                            ;mov    bx, 159h        ; disksector+EXT_BOOT.BPB
9089                                                                    ; point to the bpb in the boot record
9090                                            ;mov    al, [cs:bx+10] ; [bx+EBPB.MEDIADESCRIPTOR]
9091                                            mov     al, [bx+10] ; 12/08/2023
9092                                                                    ; get the mediadescriptor byte
9093                                            and     al, 0F0h        ; mask off low nibble
9094                                            cmp     al, 0F0h        ; is high nibble = 0Fh?
9095                                            jne     short invalid_boot_record ; no, invalid boot record
9096                                            ;cmp    word [cs:bx], 512 ; [bx+EBPB.BYTESPERSECTOR]
9097                                            cmp     word [bx], 512 ; 12/08/2023
9098                                            jne     short invalid_boot_record ; invalidate non 512 byte sectors
9099
9100                                    check2_ok:                      ; yes, mediadescriptor ok.
9101                                            mov     al, [bx+2] ; 12/08/2023
9102                                            ;mov    al, [cs:bx+2]  ; now make sure that
9103                                                                    ; the sectorspercluster is
9104                                                                    ; a power of 2
9105                                                                    ;
9106                                                                    ; [bx+EBPB.SECTORSPERCLUSTER]
9107                                                                    ; get the sectorspercluster
9108                                    %endif
9109                                            ;;;
9110                                    check_2:
9111                                            ; 18/12/2023
9112                                            ; bx = disksector
9113 000024C6 8A4715                            mov     al, [bx+21]    ; [bx+EXT_BOOT.BPB+EBPB.MEDIADESCRIPTOR]
9114                                                                    ; get the mediadescriptor byte
9115 000024C9 24F0                              and     al, 0F0h        ; mask off low nibble
9116 000024CB 3CF0                              cmp     al, 0F0h        ; is high nibble = 0Fh?
9117 000024CD 7514                              jne     short invalid_boot_record ; no, invalid boot record
9118 000024CF 817F0B0002                        cmp     word [bx+11], 512 ; [bx+EXT_BOOT.BPB+EBPB.BYTESPERSECTOR]
9119 000024D4 750D                              jne     short invalid_boot_record ; invalidate non 512 byte sectors
9120
9121                                    check2_ok:  ; yes, mediadescriptor ok.
9122 000024D6 8A470D                            mov     al, [bx+13]    ; now make sure that
9123                                                                    ; the sectorspercluster is
9124                                                                    ; a power of 2
9125                                                                    ;
9126                                                                    ; [bx++EXT_BOOT.BPB+EBPB.SECTORSPERCLUSTER]
9127                                                                    ; get the sectorspercluster
9128                                            ;;;
9129
9130 000024D9 08C0                              or      al, al         ; is it zero?
9131 000024DB 7406                              jz      short invalid_boot_record ; yes, invalid boot record
9132
9133                                    ck_power_of_two:
9134 000024DD D0E8                              shr     al, 1          ; shift until first bit emerges
9135 000024DF 73FC                              jnc     short ck_power_of_two
9136 000024E1 7406                              jz      short valid_boot_record
9137
9138                                    invalid_boot_record:
9139                                            ; 18/12/2023
9140                                            ;pop    ax
9141                                            ;pop    bx ; +
9142 000024E3 E96001                            jmp     unknown        ; jump to invalid boot record
9143                                                                    ; unformatted or illegal media.
9144                                    ; 16/12/2023
9145                                    ; ---------------------------------------------------------------------------
9146                                    ;   ; 12/08/2023
9147                                    ;setret_brdg:
9148                                    ;           jmp     setret
9149                                    ; ---------------------------------------------------------------------------
9150
9151                                    unknown3_0_j:
9152 000024E6 E96101                            jmp     unknown3_0     ; legally formatted media,
9153                                                                    ; although, content might be bad.
9154                                    ; ---------------------------------------------------------------------------
9155
9156                                    valid_boot_record:
9157                                            ; 18/12/2023
9158                                            ;pop    ax
9159                                            ;pop    bx ; +
9160
9161                                            ; 18/12/2023
9162                                            ; bx = offset disksector ; +
9163
9164                                    ; Signature found. Now check version.
9165
9166                                            ; 14/08/2023
9167 000024E9 817F08322E                        cmp     word [bx+8], '2.'
9168                                            ;cmp    word [cs:bx+8], '2.' ; 03/10/2022 (NASM syntax)
9169                                            ;;cmp   word ptr cs:[bx+8], 2E32h ; '2.'
9170 000024EE 7506                              jne     short try5
9171 000024F0 807F0A30                          cmp     byte [bx+10], '0'
9172                                            ;cmp    byte [cs:bx+0Ah], '0' ; 03/10/2022 (NASM syntax)
9173                                            ;;cmp   byte ptr cs:[bx+0Ah], 30h ; '0'
9174                                            ; 12/08/2023
9175                                            ;jnz    short try5
9176                                            ;jmp    short copybpb
9177 000024F4 7425                              je      short copybpb
9178
```

```
                                ;; ---------------------------------------------------------------------------
                                ;;  ; 12/08/2023
                                ;;setret_brdg:
                                ;;          jmp     setret
                                ;; ---------------------------------------------------------------------------
                                ;
                                ;unknown3_0_j:
                                ;           jmp     unknown3_0      ; legally formatted media,
                                ;                                   ; although, content might be bad.
                                ; ---------------------------------------------------------------------------

                                try5:
000024F6 E83B02                             call    cover_fdisk_bug

                                ; see if it is an os2 signature

                                            ; 12/08/2023
                                            ; ds = cs = BIOSDATA segment
000024F9 817F08302E                         cmp     word [bx+8], '0.'
                                            ;cmp    word [cs:bx+8], '0.' ; 03/10/2022 (NASM syntax)
                                            ;;cmp   word ptr cs:[bx+8], 2E30h ; '0.'
000024FE 750C                               jne     short no_os2
00002500 8A4707                             mov     al, [bx+7] ; 12/08/2023
                                            ;mov    al, [cs:bx+7]  ; 17/10/2022 (NASM syntax)
00002503 2C31                               sub     al, '1'
                                            ;sub    al, 31h        ; '1'
00002505 24FE                               and     al, 0FEh
00002507 7412                               jz      short copybpb  ; accept either       '1' or '2'
00002509 E93A01                             jmp     unknown
                                ; ---------------------------------------------------------------------------

                                ; no os2 signature, this is to check for real dos versions

                                no_os2:
                                            ; 12/08/2023
                                            ; ds = cs = BIOSDATA
0000250C 817F08332E                         cmp     word [bx+8], '3.'
                                            ;cmp    word [cs:bx+8], '3.' ; 03/10/2022 (NASM syntax)
                                            ;;cmp   word ptr cs:[bx+8], 2E33h ; '3.'
00002511 72D3                               jb      short unknown3_0_j ; must be 2.1 boot record.
                                                                       ; do not trust it, but still legal.
00002513 7506                               jnz     short copybpb  ; honor os2 boot record
                                                                   ; or dos 4.0 version
00002515 807F0A31                           cmp     byte [bx+10], '1' ; 12/08/2023
                                            ;cmp    byte [cs:bx+10], '1'
                                            ;;cmp   byte ptr cs:[bx+0Ah], 31h ; '1'
00002519 72CB                               jb      short unknown3_0_j ; if version >= 3.1, then o.k.
                                copybpb:

                                ; 03/10/2022

                                ; we have a valid boot sector. use the bpb in it to build the
                                ; bpb in bios. it is assumed that only
                                ;   BDS_BPB.BPB_SECTORSPERCLUSTER
                                ;   BDS_BPB.BPB_ROOTENTRIES, and
                                ;   BDS_BPB.BPB_SECTORSPERFAT
                                ; need to be set (all other values in already). fbigfat is also set.

                                ; if it is non fat based system, then just copy the bpb from the boot sector
                                ; into the bpb in bds table, and also set the boot serial number, volume id,
                                ; and system id according to the boot record.
                                ; for the non_fat system, don't need to set the other value. so just do goodret.

                                            ; 17/12/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
                                            ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2787h)
                                            ;;;
                                            ; 17/12/2023
0000251B BE[5D01]                           mov     si, disksector+11
                                            ;sub    ch, ch ; ; (ch may be > 0)
0000251E 29C9                               sub     cx, cx ; 0
                                            ;mov    cl, [disksector+16] ; BPB_NumFATs
00002520 8A4C05                             mov     cl, [si+5] ; number of FATs

                                            ; NOTE: This check is not proper for FAT32 boot sector (standard spec)
                                            ; (after PCDOS 7.1). So, it is not existing in Windows ME IO.SYS
                                            ; Erdogan Tan - 01/09/2023 ((IBMBIO.COM 7.1 disassembly note))

                                            ;;cmp   word ptr cs:disksector+4Dh, 0 ; ???
                                            ;cmp    word [disksector+4Dh], 0
                                            ;jnz    short check_3

                                            ; 17/12/2023
                                            ; check extended boot signature (0x29)
                                            ;
                                            ; (***) NOTE: 28 bytes of FAT16/FAT12 boot sector from offset 36
                                            ; have been moved to offset 64 (see label 'check_1_ok:' above)
                                            ; ((now, BS_BootSig is at offset 66 even if it was at offset 38))

                                            ;cmp    cs:disksector+42h, 29h; BS_BootSig (FAT32)
00002523 803E[9401]29                       cmp     byte [disksector+42h], 29h ; BS_BootSig (***)
                                            ;jmp    short check_4
                                check_3:
                                            ;;cmp   cs:disksector+26h, 29h; BS_BootSig (FAT16/FAT12)
                                            ;cmp    byte [disksector+26h], 29h ; (***)
                                check_4:
00002528 7538                               jnz     short copybpb_fat     ; conventional fat system

                                ; 17/12/2023
                                %if 0
                                            ; 10/12/2022
                                            ; (number of FATs optimization)
                                            mov     si, disksector+11 ; disksector+0Bh
                                            ;;mov   cl, [cs:disksector+10h] ; Number of FATs (may be 2 or 1)
                                            ;mov    cl, [cs:si+05h]
                                            ; 12/08/2023
                                            ; ds = cs = BIOSDATA segment (0070h)
                                            mov     cl, [si+05h] ; number of FATs

                                            cmp     byte [si+1Bh], 29h ; 12/08/2023
                                            ;cmp    byte [cs:si+1Bh], 29h ; 10/12/2022
                                            ;;cmp   byte [cs:disksector+26h], 29h ; 17/10/2022
                                                                   ; [disksector+EXT_BOOT.SIG]
                                                                   ; EXT_BOOT_SIGNATURE
                                            jnz     short copybpb_fat ; conventional fat system

                                            ; 03/10/2022
                                            ; 29/12/2018 - Retro DOS v4.0 modification note:
                                            ; Regarding 'fat_big_small' part of this (MSDOS 6.0) code
                                            ;           number of FATs must be 2 ; =*?=
                                            ; (Otherwise, '# of data sectors' would be calculated as wrong!!!)
                                            ;
                                            ;cmp    byte [disksector+EXT_BOOT.BPB+EBPB.NUMBEROFFATS], 2 ; =*?=

                                            ; 10/12/2022
```

```
                                        ;cmp    byte [cs:disksector+10h], 0
                                                                ; [disksector+EXT_BOOT.BPB+EBPB.NUMBEROFFATS]
                                        ;jnz    short copybpb_fat ; a fat system.
                                        or      cl, cl   ; [cs:disksector+10h]
                                        jnz     short copybpb_fat ; a fat system.
                                %endif

                                        ; 17/12/2023 - Retro DOS v5.0
                                        ;;cmp   byte [cs:disksector+10h], 0 ; BPB.fats
                                        ;cmp    byte [disksector+10h], 0 ; BPB_NUmFATs
                                        ;jnz    short copybpb_fat ; a fat system
                                        ; 17/12/2023
                                        ; cl = [disksector+10h]
0000252A 20C9                           and     cl, cl ; 0 ?
0000252C 7534                           jnz     short copybpb_fat ; a fat system

                                ; non fat based    media.

0000252E 57                             push    di  ; BDS
                                        ; 12/08/2023
                                        ;push   ds  ; ds = cs = BIOSDATA segment

                                        ; 17/12/2023
                                        ; es = ds = cs
                                        ;push   ds
                                        ;pop    es

                                        ; 12/08/2023
                                        ; ds = cs
                                        ;push   cs
                                        ;pop    ds

                                        ; 10/12/2022
                                        ; (number of FATs optimization)
                                        ; SI = disksector+11
                                        ; 17/10/2022
                                        ;;mov   si, 159h       ; disksector+EXT_BOOT.BPB
                                        ;mov    si, disksector+11
0000252F 83C706                         add     di, 6           ; add di,BDS.BPB

                                ; just for completeness, we'll make sure that total_sectors and
                                ; big_total_sectors aren't both zero. I've seen examples of
                                ; this on DOS 3.30 boot records. I don't know exactly how it
                                ; got that way. If it occurs, then use the values from the
                                ; partition table.

                                        ; 17/12/2023
                                        ; cx = 0
                                        ; 18/12/2022
                                        ;sub    cx, cx

                                        ;cmp    word [cs:si+8], 0     ; [cs:si+EBPB.TOTALSECTORS]
                                        ;jnz    short already_nonz
                                        ;                       ; how about big_total?
                                        ;cmp    word [cs:si+15h], 0   ; [cs:si+EBPB.BIGTOTALSECTORS]
                                        ;jnz    short already_nonz ; we're okay if any are != 0
                                        ;cmp    word [cs:si+17h], 0   ; [cs:si+EBPB.BIGTOTALSECTORS+2]
                                        ;jnz    short already_nonz

                                        ; 12/08/2023
                                        ; ds = cs = BIOSDATA segment (0070h)

                                        ; 17/12/2023
                                        ; 12/08/2023
00002532 394C08                         cmp     [si+8], cx ; 0        ; [si+EBPB.TOTALSECTORS]
00002535 751C                           jnz     short already_nonz
                                                                ; how about big_total?
00002537 394C15                         cmp     [si+15h], cx ; 0      ; [si+EBPB.BIGTOTALSECTORS]
0000253A 7517                           jnz     short already_nonz ; we're okay if any are != 0
0000253C 394C17                         cmp     [si+17h], cx ; 0      ; [si+EBPB.BIGTOTALSECTORS+2]
0000253F 7512                           jnz     short already_nonz

                                ; now let's copy the values from the partition table (now in the BDS)
                                ; into the BPB in the boot sector buffer, before they get copied back.

00002541 8B4508                         mov     ax, [di+8]     ; [di+BDS.totalsecs16]
                                        ; 12/08/2023
                                        ;mov    [cs:si+8], ax  ; [cs:si+EBPB.TOTALSECTORS]
00002544 894408                         mov     [si+8], ax
00002547 8B4515                         mov     ax, [di+15h]   ; [di+BDS.totalsecs32]
                                        ;mov    [cs:si+15h], ax       ; [cs:si+EBPB.BIGTOTALSECTORS]
0000254A 894415                         mov     [si+15h], ax
0000254D 8B4517                         mov     ax, [di+17h]   ; [di+BDS.totalsecs32+2]
                                        ;mov    [cs:si+17h], ax       ; [cs:si+EBPB.BIGTOTALSECTORS+2]
00002550 894417                         mov     [si+17h], ax

                                already_nonz:
                                        ; 18/12/2022
                                        ; cx = 0
                                        ;mov    cl, 25
                                        ;;mov   cx, 25         ; A_BPB.size - 6 ; Use SMALL version!
                                        ; 17/12/2023 - Retro DOS v5.0
00002553 B135                           mov     cl, 53  ; PCDOS 7.1 IBMBIO.COM
                                                        ; BDS.BPB size (25 + 28 for FAT32 parms)
00002555 F3A4                           rep movsb
                                        ;pop    ds
                                        ; 12/08/2023
                                        ; ds = cs
                                        ;pop    bp  ; ds (on top of stack) = BIOSDATA
00002557 5F                             pop     di  ; BDS
                                        ;push   es
                                        ;push   ds
                                        ;pop    es
                                        ;push   cs
                                        ;pop    ds
                                        ; 12/08/2023
                                        ;mov    es, bp
                                        ; ds = cs = es

                                        ; 14/08/2023
00002558 BD[4F08]                       mov     bp, MOVMEDIAIDS ; mov_media_ids
                                        ; 18/12/2022
                                        ;mov    bp, mov_media_ids
                                        ;;mov   bp, 751h       ; mov_media_ids
                                                                ; at 2C7h:751h = 70h:2CC1h
                                                                ; set volume id, systemid, serial.
0000255B 0E                             push    cs              ; simulate far call
0000255C E812F5                         call    call_bios_code
                                        ; 12/08/2023
                                        ; ds = cs = es
                                        ;push   es
                                        ;pop    ds
                                        ;pop    es
0000255F E9C701                         jmp     goodret
```

```
9427
9428                                    ; ----------------------------------------------------------------------------
9429
9430                                    ; ****** cas ---
9431                                    ; IBM DOS 3.30 doesn't seem to mind that the TOTAL_SECTORS and
9432                                    ; BIG_TOTAL_SECTORS field in the boot sector are 0000. This
9433                                    ; happens with some frequency -- perhaps through some OS/2 setup
9434                                    ; program. We haven't actually been COPYING the TOTAL_SECTORS
9435                                    ; from the boot sector into the DPB anyway, we've just been using
9436                                    ; it for calculating the fat size. Pretty scary, huh? For now,
9437                                    ; we'll go ahead and copy it into the DPB, except in the case
9438                                    ; that it equals zero, in which case we just use the values in
9439                                    ; the DPB from the partition table.
9440
9441                                    ; 17/10/2022
9442                                    ;MOVMEDIAIDS equ mov_media_ids - DOSBIOSEG_2C7h ; (751h for MSDOS 5.0 IO.SYS)
9443                                    ;CLEARIDS equ clear_ids - DOSBIOSEG_2C7h ; (5D9h for MSDOS 5.0 IO.SYS)
9444                                    ; 09/12/2022
9445                                    MOVMEDIAIDS equ mov_media_ids
9446                                    CLEARIDS equ clear_ids
9447                                    ; 11/09/2023
9448                                    CLEARIDS_X equ clear_ids_x
9449
9450                                    copybpb_fat:
9451                                                ; 17/12/2023
9452                                                ; ch = 0, cl = number of FATs
9453                                                ; 10/12/2022
9454                                                ; (number of FATs optimization)
9455                                                ; SI = disksector+11
9456                                                ; 17/10/2022
9457                                                ;mov    si, disksector+11
9458                                                ;;mov   si, 159h       ; disksector+EXT_BOOT.BPB
9459                                                                      ; cs:si -> bpb in boot
9460                                                ; 17/12/2023
9461                                                ; dx = 0
9462                                                ; 18/04/2024 (BugFix)
9463 00002562 31D2                                  xor     dx, dx
9464
9465                                                ; 12/08/2023
9466                                                ; ds = cs = BIOSDATA segment (0070h)
9467 00002564 8B4408                               mov     ax, [si+8]
9468                                                ;mov    ax, [cs:si+8]  ; [cs:si+EBPB.TOTALSECTORS]
9469                                                                       ; get totsec from boot sec
9470 00002567 09C0                                 or      ax, ax
9471 00002569 7514                                 jnz     short copy_totsec ; if non zero, use that
9472 0000256B 8B4415                               mov     ax, [si+15h] ; 12/08/2023
9473                                                ;mov    ax, [cs:si+15h]       ; [cs:si+EBPB.BIGTOTALSECTORS]
9474                                                                       ; get the big version
9475                                                                       ; (32 bit total       sectors)
9476 0000256E 8B5417                               mov     dx, [si+17h] ; 12/08/2023
9477                                                ;mov    dx, [cs:si+17h]       ; [cs:si+EBPB.BIGTOTALSECTORS+2]
9478                                                ; 10/12/2022
9479                                                ; (number of FATs optimization)
9480                                                ; CL = number of FATs (2 or 1)
9481 00002571 89D3                                 mov     bx, dx         ; see if it is a big zero
9482 00002573 09C3                                 or      bx, ax
9483 00002575 7508                                 jnz     short copy_totsec
9484                                                                       ; screw it. it was bogus.
9485 00002577 8B451B                               mov     ax, [di+1Bh]   ; [di+BDS.totalsecs32]
9486 0000257A 8B551D                               mov     dx, [di+1Dh]   ; [di+BDS.totalsecs32+2]
9487 0000257D EB06                                 jmp     short fat_big_small
9488
9489                                                ;mov    cx, dx
9490                                                ;or     cx, ax         ; see if it is a big zero
9491                                                ;jz     short totsec_already_set ; screw it. it     was bogus.
9492                                    copy_totsec:
9493 0000257F 89451B                               mov     [di+1Bh], ax   ; [di+BDS.totalsecs32]
9494                                                                       ; make DPB match boot sec
9495 00002582 89551D                               mov     [di+1Dh], dx   ; [di+BDS.totalsecs32+2]
9496
9497                                                ; 10/12/2022
9498                                    ;totsec_already_set:
9499                                                ;mov    ax, [di+1Bh]   ; [di+BDS.totalsecs32]
9500                                                ;mov    dx, [di+1Dh]   ; [di+BDS.totalsecs32+2]
9501
9502                                    ; determine fat entry size.
9503
9504                                    fat_big_small:
9505
9506                                    ;at this moment dx;ax = total sector number
9507
9508                                    ;Do not assume 1 reserved sector. Update the reserved sector field in BDS
9509                                    ;from the BPB on the disk
9510
9511                                                ; 12/08/2023
9512                                                ; ds = cs = BIOSDATA segment (0070h)
9513
9514 00002585 8B5C03                               mov     bx, [si+3]
9515                                                ;mov    bx, [cs:si+3]  ; [cs:si+EBPB.RESERVEDSECTORS]
9516                                                                       ; get #reserved_sectors       from BPB
9517 00002588 895D09                               mov     [di+9], bx     ; [di+BDS.resectors]
9518                                                                       ; update BDS field
9519 0000258B 29D8                                 sub     ax, bx
9520 0000258D 83DA00                               sbb     dx, 0          ; update the count
9521                                                ; 12/08/2023
9522 00002590 8B5C0B                               mov     bx, [si+0Bh]
9523                                                ;mov    bx, [cs:si+0Bh]       ; [cs:si+EBPB.SECTORSPERFAT]
9524                                                                       ; bx = sectors/fat
9525 00002593 895D11                               mov     [di+11h], bx   ; [di+BDS.fatsecs]
9526                                                                       ; set in bds bpb
9527                                                ; 17/12/2023 - Retro DOS v5.0
9528                                                ;            (PCDOS 7.1 IBMBIO.COM)
9529 00002596 53                                   push    bx ; FAT sectors
9530 00002597 09DB                                 or      bx, bx
9531 00002599 753A                                 jnz     short fat_16bit
9532
9533                                    ; 17/12/2023
9534                                    %if 0
9535                                                sub     ax, [si+19h]   ; FAT32 file system (PCDOS 7.1 BUG!)
9536                                                                       ; BPB.FATSz32
9537                                                sbb     dx, [si+1Bh]   ; BPB.FATSz32+2 (PCDOS 7.1 BUG!)
9538                                                ; dx:ax = partition size - (one FAT sectors + reserved secta)
9539                                                mov     bx, [si+19h]   ; BPB.FATSz32
9540                                                mov     [di+1Fh], bx   ; [di+BDS.fatsecs32]
9541                                                mov     bx, [si+1Bh]   ; BPB.FATSz32+2
9542                                                mov     [di+21h], bx   ; [di+BDS.fatsecs32+2]
9543                                                mov     bx, [si+1Dh]   ; BPB.BPB_ExtFlags
9544                                                mov     [di+23h], bx   ; [di+BDS.extflags]
9545                                                mov     bx, [si+1Fh]   ; BPB.FSVer
9546                                                mov     [di+25h], bx   ; [di+BDS.fsver]
9547                                                mov     bx, [si+21h]   ; BPB.RootClus
9548                                                mov     [di+27h], bx   ; [di+BDS.rootdirclust]
9549                                                mov     bx, [si+23h]   ; BPB.RootClus+2
9550                                                mov     [di+29h], bx   ; [di+BDS.rootdirclust+2]
```

```
9551                                            mov     bx, [si+25h]   ; BPB.FSInfo
9552                                            mov     [di+2Bh], bx   ; [di+BDS.fsinfo]
9553                                            mov     bx, [si+27h]   ; BPB.FSInfo+2
9554                                            mov     [di+2Dh], bx   ; [di+BDS.fsinfo+2]
9555                                            jmp     short fat_32bit       ; PCDOS 7.1 BUG! Erdogan Tan - 8/8/2023
9556                                                                          ; correct code (would be):
9557                                                                          ;   mov cl, [cs:si+05h] ; BPB_NumFATs
9558                                                                          ; sub_fat32_size:
9559                                                                          ;   sub ax, [cs:si+19h] ; BPB_FATSz32
9560                                                                          ;   sbb dx, [cs:si+1Bh] ; BPB_FATSz32+2
9561                                                                          ;   dec cl
9562                                                                          ;   jg short sub_fat32_size
9563                                                                          ;   jmp short fat_32bit
9564                                    %endif
9565                                            ; 17/12/2023
9566                                            ; cl = BPB_NumFATs (2 or 1)
9567                                            ; ch = 0
9568 0000259B 8B5C19                           mov     bx, [si+19h]   ; BPB.FATSz32
9569                                    sub_fat32_size:
9570 0000259E 29D8                             sub     ax, bx
9571 000025A0 1B541B                           sbb     dx, [si+1Bh]   ; BPB.FATSz32+2
9572                                            ;dec    cl
9573 000025A3 49                               dec     cx
9574 000025A4 7FF8                             jg      short sub_fat32_size
9575
9576 000025A6 895D1F                           mov     [di+1Fh], bx   ; [di+BDS.fatsecs32]
9577 000025A9 8B5C1B                           mov     bx, [si+1Bh]   ; BPB.FATSz32+2
9578 000025AC 895D21                           mov     [di+21h], bx   ; [di+BDS.fatsecs32+2]
9579
9580 000025AF 8B5C1D                           mov     bx, [si+1Dh]   ; BPB.BPB_ExtFlags
9581 000025B2 895D23                           mov     [di+23h], bx   ; [di+BDS.extflags]
9582 000025B5 8B5C1F                           mov     bx, [si+1Fh]   ; BPB.FSVer
9583 000025B8 895D25                           mov     [di+25h], bx   ; [di+BDS.fsver]
9584 000025BB 8B5C21                           mov     bx, [si+21h]   ; BPB.RootClus
9585 000025BE 895D27                           mov     [di+27h], bx   ; [di+BDS.rootdirclust]
9586 000025C1 8B5C23                           mov     bx, [si+23h]   ; BPB.RootClus+2
9587 000025C4 895D29                           mov     [di+29h], bx   ; [di+BDS.rootdirclust+2]
9588 000025C7 8B5C25                           mov     bx, [si+25h]   ; BPB.FSInfo
9589 000025CA 895D2B                           mov     [di+2Bh], bx   ; [di+BDS.fsinfo]
9590 000025CD 8B5C27                           mov     bx, [si+27h]   ; BPB.FSInfo+2
9591 000025D0 895D2D                           mov     [di+2Dh], bx   ; [di+BDS.fsinfo+2]
9592 000025D3 EB08                             jmp     short fat_32bit
9593
9594                                    fat_16bit:
9595                                            ; 17/12/2023 - Retro DOS v5.0
9596                                            ;            (PCDOS 7.1 IBMBIO.COM)
9597                                            ; 10/12/2022
9598                                            ; (number of FATs optimization)
9599                                            ; CL = number of FATs (2 or 1)
9600                                            ; CH = 0 ; 17/12/2023
9601                                            ;dec    cl ; *
9602                                            ; 18/12/2022
9603 000025D5 49                               dec     cx ; *
9604 000025D6 D3E3                             shl     bx, cl
9605                                            ;shl    bx, 1   ; =*?=  ; always 2 fats
9606
9607 000025D8 29D8                             sub     ax, bx         ; sub # fat sectors
9608 000025DA 83DA00                           sbb     dx, 0
9609                                    fat_32bit:
9610                                            ; 17/12/2023
9611 000025DD 8B5C06                           mov     bx, [si+6] ; 12/08/2023
9612                                            ;mov    bx, [cs:si+6]  ; [cs:si+EBPB.ROOTENTRIES]
9613                                                                   ; # root entries
9614 000025E0 895D0C                           mov     [di+0Ch], bx   ; [di+BDS.direntries]
9615                                                                   ; set in bds bpb
9616 000025E3 B104                             mov     cl, 4
9617 000025E5 D3EB                             shr     bx, cl         ; div by 16 ents/sector
9618 000025E7 29D8                             sub     ax, bx         ; sub # dir sectors
9619 000025E9 83DA00                           sbb     dx, 0          ;
9620                                                                   ; dx:ax now contains the
9621                                                                   ; # of data sectors
9622                                            ; 17/12/2023
9623                                            ; ch = 0
9624                                            ;xor    cx, cx ; *
9625 000025EC 8A4C02                           mov     cl, [si+2] ; 12/08/2023
9626                                            ;mov    cl, [cs:si+2]  ; [cs:si+EBPB.SECTORSPERCLUSTER]
9627                                                                   ; sectors per cluster
9628 000025EF 884D08                           mov     [di+8],cl      ; [di+BDS.secperclus]
9629                                                                   ; set in bios bpb
9630 000025F2 50                               push    ax
9631 000025F3 89D0                             mov     ax, dx
9632 000025F5 31D2                             xor     dx, dx
9633 000025F7 F7F1                             div     cx             ; cx = sectors per cluster
9634                                            ; 12/08/2023 (ds=cs)
9635                                            ;mov    [temp_h], ax
9636                                            ;;mov   [cs:temp_h], ax       ; [temp_h]:ax now contains the
9637                                                                   ; # clusters.
9638                                            ; 17/12/2023
9639 000025F9 A3[9E04]                         mov     [saved_word], ax ; hw of cluster number
9640 000025FC 58                               pop     ax
9641 000025FD F7F1                             div     cx
9642                                            ; 17/12/2023
9643                                            ;;cmp   word [cs:temp_h], 0
9644                                            ;cmp    word [temp_h], 0  ; 12/08/2023
9645                                            ;cmp    word [saved_word], 0 ; (*)
9646                                            ;ja     short toobig_ret ; too big cluster number
9647
9648                                            ; 17/12/2023
9649                                            ;;;
9650 000025FF 5B                               pop     bx ; FAT sectors (16 bit)
9651                                            ;and    bx, bx ; 0 ?
9652 00002600 09DB                             or      bx, bx ; 0 ?
9653 00002602 751F                             jnz     short chk_clnum_hw
9654                                                            ; 16 bit fat sectors > 0 ; FAT12 or FAT16 fs
9655
9656 00002604 813E[9E04]FF0F                   cmp     word [saved_word], 0FFFh
9657 0000260A 7503                             jne     short fat32_clust_limit
9658 0000260C 83F8F6                           cmp     ax, 0FFF6h     ; FAT32 cluster number limit: 0FFFFFF6h
9659                                    fat32_clust_limit:
9660 0000260F 772D                             ja      short short toobig_ret ; too big cluster number
9661 00002611 391E[9E04]                       cmp     [saved_word], bx ; 0 ?
9662                                            ;jnz    short fat16_clust_limit
9663 00002615 7505                             jnz     short set_bigbig_flag ; 17/12/2023
9664                                    fat16_clust_limit: ; 17/12/2023
9665 00002617 83F8F6                           cmp     ax, 0FFF6h     ; FAT16 cluster number limit: 0FFF6h
9666                                    ;fat16_clust_limit:
9667 0000261A 760E                             jna     short fat12_clust_limit ; jbe
9668                                    set_bigbig_flag:   ; 17/12/2023
9669 0000261C 800E[061A]20                     or      byte [fbigfat], 20h ; fbigbig ; FAT32 fs
9670 00002621 EB11                             jmp     short copymediaid
9671                                    chk_clnum_hw:
9672 00002623 833E[9E04]00                     cmp     word [saved_word], 0 ; (*)
9673 00002628 7714                             ja      short toobig_ret ; too big cluster number
9674                                            ;;;
```

```
                                fat12_clust_limit:
0000262A 3DF60F                         cmp     ax, 0FF6h       ; 4096-10
                                                                ; is this 16-bit fat?
0000262D 7205                           jb      short copymediaid ; no,       small fat
                                        ; 17/10/2022
0000262F 800E[061A]40                   or      byte [fbigfat], 40h ; fbig ; FAT16 fs
                                        ;or     ds:fbigfat, 40h         ; fbig
                                                                ; 16 bit fat
                                copymediaid:
                                        ; 17/12/2023
                                        ; es = ds = cs

                                        ;push   es
                                        ;push   ds
                                        ;pop    es

                                        ; 12/08/2023
                                        ; ds = cs = BIOSDATA
                                        ;push   cs
                                        ;pop    ds
                                        ; 17/10/2022
00002634 BD[4F08]                       mov     bp, MOVMEDIAIDS
                                        ;mov    bp, 865h        ; (PCDOS 7.1 IBMBIO.COM)
                                        ;;mov   bp, 751h        ; mov_media_ids
                                                                ; at 2C7h:751h = 70h:2CC1h
                                                                ; copy filesys_id, volume label
00002637 0E                             push    cs              ; simulate far call
00002638 E836F4                         call    call_bios_code

                                        ; 12/08/2023
                                        ;push   es
                                        ;pop    ds
                                        ; 17/12/2023
                                        ;pop    es

0000263B E9CD00                         jmp     massage_bpb     ; now final check for bpb info
                                                                ; and return.
                                ; ---------------------------------------------------------------------------

                                toobig_ret:
                                        ; 12/08/2023 (ds=cs=BIOSDATA)
0000263E 800E[061A]80                   or      byte [fbigfat], 80h ; ftoobig
                                        ;or     byte [cs:fbigfat], 80h ; ftoobig
                                                                ; too big (32 bit clust #) for FAT16
00002643 E9E300                         jmp     goodret         ; still drive letter is       assigned
                                                                ; but useless. to big for
                                                                ; current pc dos fat file system
                                ; ---------------------------------------------------------------------------

                                unknown:
                                        ; 17/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
00002646 804D4002                       or      byte [di+40h], 2 ; [di+BDS.flags+1]
                                                                 ; unformatted_media
                                        ; 12/12/2022
                                        ;or     byte [di+24h], 02h
                                        ;;or    word [di+23h], 200h ; [di+BDS.flags]
                                                                ; unformatted_media
                                                                ; Set unformatted media       flag.

                                ; the boot signature may not be   recognizable,
                                ; but we should    try and read it anyway.

                                unknown3_0:
0000264A 8B551D                         mov     dx, [di+1Dh]    ; skip setting unformatted_media bit
                                                                ; [di+BDS.totalsecs32+2]
0000264D 8B451B                         mov     ax, [di+1Bh]    ; [di+BDS.totalsecs32]
00002650 BE[161A]                       mov     si, disktable2
                                scan:                                   ; 08/08/2023
                                        ;cmp    dx, [cs:si]     ; total sectors hw
                                        ; 12/08/2023 (ds=cs)
00002653 3B14                           cmp     dx, [si]
00002655 720C                           jb      short gotparm
00002657 7705                           ja      short scan_next
                                        ;cmp    ax, [cs:si+2]   ; total sectors lw
00002659 3B4402                         cmp     ax, [si+2]
0000265C 7605                           jbe     short gotparm
                                scan_next:
0000265E 83C60A                         add     si, 10          ; 5*2
00002661 EBF0                           jmp     short scan      ; covers upto 512 mb media
                                ; ---------------------------------------------------------------------------

                                gotparm:
00002663 8A4C08                         mov     cl, [si+8]      ; fat size for fbigfat flag
                                        ;or     ds:fbigfat, cl
                                        ; 17/10/2022
00002666 080E[061A]                     or      [fbigfat], cl   ; (fbig flag, 40h or 0) ; 08/08/2023
                                                                ; 12/08/2023
                                        ; ds = cs = BIOSDATA
0000266A 8B4C04                         mov     cx, [si+4]
                                        ;mov    cx, [cs:si+4]   ; ch = number of sectors per cluster
                                                                ; cl = log base 2 of ch
0000266D 8B5406                         mov     dx, [si+6]
                                        ;mov    dx, [cs:si+6]   ; dx = number of root dir entries

                                ; now calculate size of fat table

00002670 89550C                         mov     [di+0Ch], dx    ; [di+BDS.direntries]
                                                                ; save number of (root)       dir entries
00002673 8B551D                         mov     dx, [di+1Dh]    ; [di+BDS.totalsecs32+2]
00002676 8B451B                         mov     ax, [di+1Bh]    ; [di+BDS.totalsecs32]
00002679 886D08                         mov     [di+8], ch      ; [di+BDS.secperclus]
                                                                ; save sectors per cluster

                                        ; 17/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
0000267C F606[061A]60                   test    byte [fbigfat], 60h ; fbig+fbigbig ; FAT16 or FAT32
                                        ; 11/09/2023
                                        ; 17/10/2022
                                        ;test   byte [fbigfat], 40h
                                        ;;test  ds:fbigfat, 40h         ; fbig
                                                                ; if (fbigfat)
00002681 751E                           jnz     short dobig     ; goto dobig; (16 bit fat)

                                ; we don't need to change "small fat" logic since it is guaranteed
                                ; that double word total sector will not use 12 bit fat (unless
                                ; it's sectors/cluster >= 16 which will never be in this case.)
                                ; so in this case we assume dx = 0 !!

00002683 31DB                           xor     bx, bx          ; 12 bit fat (FAT12 fs)
00002685 88EB                           mov     bl, ch
00002687 4B                             dec     bx
00002688 01C3                           add     bx, ax          ; dx=0
0000268A D3EB                           shr     bx, cl          ; bx = 1+(bpb->maxsec+BDS.secperclus-1)/
0000268C 43                             inc     bx              ; BDS.secperclus
0000268D 80E3FE                         and     bl, 0FEh        ; bx &= ~1; (=number of clusters)
```

```
9799 00002690 89DE                              mov     si, bx
9800 00002692 D1EB                              shr     bx, 1
9801 00002694 01F3                              add     bx, si          ; number of FAT bytes ; 08/08/2023
9802 00002696 81C3FF01                          add     bx, 511         ; bx += 511 + bx/2
9803 0000269A D0EF                              shr     bh, 1           ; bh >>= 1; (=bx/512)
9804 0000269C 887D11                            mov     [di+11h], bh    ; [di+BDS.fatsecs]
9805                                                                    ; save number of fat sectors
9806 0000269F EB6A                              jmp     short massage_bpb
9807                                    ; ---------------------------------------------------------------------------
9808
9809                                    ; for bigfat we do need to extend this logic to 32 bit sector calculation.
9810
9811                                    dobig:
9812 000026A1 B104                              mov     cl, 4           ; 16 (2^4) directory entries per sector
9813 000026A3 52                                push    dx              ; save total sectors (high)
9814 000026A4 8B550C                            mov     dx, [di+0Ch]    ; [di+BDS.direntries]
9815 000026A7 D3EA                              shr     dx, cl          ; root dir sectors = BDS.direntries / 16;
9816 000026A9 29D0                              sub     ax, dx
9817 000026AB 5A                                pop     dx
9818 000026AC 83DA00                            sbb     dx, 0           ; dx:ax = total sectors - root dir sectors
9819 000026AF 83E801                            sub     ax, 1
9820 000026B2 83DA00                            sbb     dx, 0           ; dx:ax = t - r - d
9821                                                                    ; total secs - reserved       secs - root dir       secs
9822 000026B5 B302                              mov     bl, 2
9823 000026B7 8A7D08                            mov     bh, [di+8]      ; [di+BDS.secperclus]
9824                                                                    ; bx = 256 * BDS.secperclus + 2
9825
9826                                    ; I don't understand why to add bx here!!!
9827
9828                                                    ; 29/12/2018 - Erdogan Tan (Retro DOS v4.0)
9829                                                    ; 27/09/2022
9830                                                    ; (Microsoft FAT32 File      System Specification,
9831                                                    ; December 2000, Page 21)
9832                                                    ; TmpVal1 = DskSize - (BPB_ResvdSecCnt+RootrDirSectors)
9833                                                    ; TmpVal2 = (256*BPB_SecPerClus)+BPB_NumFATs
9834                                                    ; 8/8/2023 (Retro DOS v5.0)
9835                                                    ; If(FATType == FAT32)
9836                                                    ;   TmpVal2 = TmpVal2 / 2;
9837                                                    ; FATsz = (TmpVal1+(TmpVal2-1))/TmpVal2
9838                                                    ; 8/8/2023 (Retro DOS v5.0)
9839                                                    ; If(FATType == FAT32) {
9840                                                    ;   BPB_FATSz16 = 0;
9841                                                    ;   BPB_FATSz32 = FATSz;
9842                                                    ;} else {
9843                                                    ;   BPB_FATSz16 = LOWORD(FATSz);
9844                                                    ;/* there is no BPB_FATSz32 in a FAT16 BPB */
9845                                                    ;}
9846                                                                            ; dx:ax = TmpVal1, bx = TmpVal2
9847 000026BA 01D8                                add     ax, bx          ;
9848 000026BC 83D200                            adc     dx, 0           ; dx:ax = TmpVal1+TmpVal2
9849 000026BF 83E801                            sub     ax, 1
9850 000026C2 83DA00                            sbb     dx, 0           ; dx:ax = TmpVal1+TmpVal2-1
9851
9852                                                    ;;;
9853                                                    ; 17/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
9854 000026C5 F606[061A]20                      test    byte [fbigfat], 20h ; fbigbig (FAT32) flag
9855 000026CA 740D                              jz      short dobig1
9856
9857 000026CC D1EB                              shr     bx, 1           ; TmpVal2 = TmpVal2 / 2
9858                                                                    ; dx:ax = TmpVal1+(2*TmpVal2)-1
9859 000026CE 83E81F                            sub     ax, 31          ; reserved sectors = 32 (for FAT32 fs) /// 1+31 = 32
9860 000026D1 83DA00                            sbb     dx, 0
9861 000026D4 29D8                              sub     ax, bx
9862 000026D6 83DA00                            sbb     dx, 0           ; dx:ax = TmpVal1+(2*TmpVal2)-TmpVal2-1
9863                                                                    ;       = TmpVal1+(TmpVal2-1)
9864                                    dobig1:
9865 000026D9 50                                push    ax              ; save lw of dividend
9866 000026DA 89D0                              mov     ax, dx          ; divide hw of dx:ax at first (as 1st stage)
9867 000026DC 31D2                              xor     dx, dx
9868 000026DE F7F3                              div     bx              ; 32 bit division, dx:ax/bx
9869                                                                    ; remainder in dx is hw of 2nd stage dividend
9870 000026E0 89C5                              mov     bp, ax          ; hw of quotient
9871 000026E2 58                                pop     ax              ; restore lw of dividend (of 1st stage)
9872                                                    ;;;
9873
9874                                    ; assuming dx in the table will never be bigger than bx.
9875
9876 000026E3 F7F3                              div     bx              ; BDS.fatsecs =
9877                                                                    ; ceil((total-dir-res)/(256*BDS.secperclus+2))
9878 000026E5 894511                            mov     [di+11h], ax    ; [di+BDS.fatsecs]
9879                                                                    ; number of fat       sectors
9880                                                    ;;;
9881
9882                                                    ; 18/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
9883 000026E8 8A1E[061A]                        mov     bl, [fbigfat]
9884 000026EC 885D3B                            mov     [di+3Bh], bl    ; [di+BDS.fatsiz] ; fat size flag
9885
9886 000026EF F6C320                            test    bl, 20h         ; fbigbig (FAT32) flag
9887 000026F2 7410                              jz      short dobig2    ; not FAT32
9888
9889 000026F4 89451F                            mov     [di+1Fh], ax    ; [di+BDS.fatsecs32]
9890 000026F7 896D21                            mov     [di+21h], bp    ; [di+BDS.fatsecs32+2]
9891 000026FA C745110000                        mov     word [di+11h], 0 ; [di+BDS.fatsecs] = 0
9892                                                                    ; clear 16 bit FAT size field
9893 000026FF C745092000                        mov     word [di+9], 32         ; [di+BDS.resectors]
9894                                                                    ; set reserved sectors to 32 (FAT32 de facto)
9895                                    dobig2:
9896                                                    ;;;
9897
9898                                    ; now, set the default filesys_id, volume label, serial number
9899
9900                                                    ; 05/08/2023
9901                                                    ; [di+1Fh] = [fbigfat]
9902                                                    ;
9903                                                    ;;mov   bl, ds:fbigfat
9904                                                    ;; 17/10/2022
9905                                                    ;mov    bl, [fbigfat]
9906                                                    ;mov    [di+1Fh], bl    ; [di+BDS.fatsiz] ; fat       size flag
9907
9908                                                    ; 12/08/2023
9909                                                    ;push   ds ; ds = cs = BIOSDATA
9910
9911                                                    ; 17/12/2023
9912                                                    ; es = ds = cs
9913                                                    ;push   ds
9914                                                    ;pop    es
9915
9916                                                    ; 12/08/2023
9917                                                    ; ds = cs = BIOSDATA
9918                                                    ;push   cs
9919                                                    ;pop    ds
9920
9921                                                    ; 18/12/2023 - Retro DOS v5.0
9922                                                    ; bl = [fbigfat] (clear_ids_x uses bl value here)
```

```
                                        ; 11/09/2023
                                        ;mov    al, [fbigfat]
00002704 BD[A106]                       mov     bp, CLEARIDS_X ; clear_ids_x (uses AL value here)
                                        ; 17/10/2022
                                        ;mov    bp, CLEARIDS
                                        ;;mov   bp, 5D9h        ; clear_ids
                                                                ; at 2C7h:5D9h = 70h:2B49h
                                                                ; at BIOSCODE:06ABh
                                                                ;      in PCDOS 7.1 IBMBIO.COM
00002707 0E                             push    cs
00002708 E866F3                         call    call_bios_code

                                        ; 12/08/2023
                                        ;pop    ds ; ds = cs = BIOSDATA

                                ; at this point, in bpb of bds table, BDS_BPB.BPB_BIGTOTALSECTORS which is
                                ; set according to the partition information. we are going to
                                ; see if (hidden sectors + total sectors) > a word. if it is true,
                                ; then no change. otherwise, BDS_BPB.BPB_BIGTOTALSECTORS will be moved
                                ; to BDS_BPB.BPB_TOTALSECTORS and BDS_BPB.BPB_BIGTOTALSECTORS will be set to 0.
                                ; we don't do this for the bpb information from the boot record. we
                                ; are not going to change the bpb information from the boot record.

                                massage_bpb:
                                        ; 05/08/2023
                                        ; [di+1Fh] = [fbigfat]
                                        ;
                                        ;; 12/12/2022
                                        ;mov    bl, [fbigfat]
                                        ;mov    [di+1Fh], bl    ; [di+BDS.fatsiz]
                                        ;                       ; set size of fat on media
                                        ;
0000270B 8B551D                         mov     dx, [di+1Dh]    ; [di+BDS.totalsecs32+2]
0000270E 8B451B                         mov     ax, [di+1Bh]    ; [di+BDS.totalsecs32]
                                        ; 11/09/2023
00002711 09D2                           or      dx, dx
00002713 7514                           jnz     short goodret
                                        ;cmp    dx, 0           ; double word total sectors?
                                        ;;ja    short goodret   ; don't have to change it.
                                        ;; 12/12/2022
                                        ;ja     short short goodret2
                                        ;cmp    word [di+19h], 0 ; [di+BDS.hiddensecs+2]
                                        ;ja     short goodret  ; don't have to change it.
                                        ; 12/12/2022
00002715 395519                         cmp     [di+19h], dx ; 0
                                        ;ja     short goodret2
00002718 770F                           ja      short goodret  ; 11/09/2023
0000271A 034517                         add     ax, [di+17h]   ; [di+BDS.hiddensecs]
                                        ;jb     short goodret
                                        ; 12/12/2022
                                        ;jc     short goodret
0000271D 7209                           jc      short goodret_clc ; 11/09/2023
0000271F 8B451B                         mov     ax, [di+1Bh]   ; [di+BDS.totalsecs32]
00002722 89450E                         mov     [di+0Eh], ax   ; [di+BDS.totalsecs16]
                                        ;mov    word [di+1Bh], 0 ; [di+BDS.totalsecs32]
                                        ; 12/12/2022
00002725 89551B                         mov     [di+1Bh], dx ; 0
                                goodret_clc:
                                        ; 11/09/2023
00002728 F8                             clc
                                goodret:
                                        ;mov    bl, ds:fbigfat
                                        ; 11/09/2023
                                        ; 12/12/2022
                                        ; 17/10/2022
00002729 8A1E[061A]                     mov     bl, [fbigfat]
                                        ; 17/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
0000272D 885D3B                         mov     [di+3Bh], bl   ; [di+BDS.fatsiz]
                                        ;mov    [di+1Fh], bl   ; [di+BDS.fatsiz]
                                                               ; set size of fat on media
                                        ; 11/09/2023
                                        ;clc
                                ret_hard_err:
                                        ; 12/12/2022
                                goodret2:
00002730 07                             pop     es
                                        ;pop    ds      ; ds = cs = BIOSDATA ; 14/08/2023
00002731 5B                             pop     bx
00002732 5F                             pop     di
00002733 C3                             retn

                                ; =============== S U B   R O U T I N E =======================================

                                ; 15/10/2022

                                ;fdisk of pc dos 3.3 and below, os2 1.0 has a bug. the maximum number of
                                ;sector that can be handled by pc dos 3.3 ibmbio should be 0ffffh.
                                ;instead, sometimes fdisk use 10000h to calculate the maximum number.
                                ;so, we are going to check that if BPB_TOTALSECTORS + hidden sector = 10000h
                                ;then subtract 1 from BPB_TOTALSECTORS.

                                        ; 17/10/2022
                                cover_fdisk_bug:
                                        ; 12/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                        ; ds = cs

                                        ; 18/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                        ; (optimization)
                                        ;push   ax
                                        ;push   dx
                                        ;push   si

                                        ; 18/12/2023
                                        ; bx = offset disksector

                                        ; 18/04/2024 - Retrodos v5.0 (PCDOS 7.1 IBMBIO.COM)
                                        ;;cmp   word [cs:disksector+16h],0
                                        ;cmp    word [disksector+16h],0 ; BPB_FATSz16
00002734 837F1600                       cmp     word [bx+16h],0
00002738 742C                           jz      short cfb_retit ; FAT32 boot sector

                                        ; 18/12/2023
0000273A 807F2629                       cmp     byte [bx+26h], 29h
                                        ; 12/08/2023
                                        ;cmp    byte [disksector+26h], 29h
                                        ;;cmp   byte [cs:disksector+26h], 29h
                                                                ; [disksector+EXT_BOOT.SIG],
                                                                ; EXT_BOOT_SIGNATURE
0000273E 7426                           je      short cfb_retit       ; if extended bpb, then       >= pc dos 4.00

00002740 817F073130                     cmp     word [bx+7], 3031h
                                        ;cmp    word [cs:bx+7], 3031h ; '10' ; os2 1.0 = ibm 10.0
00002745 7506                           jne     short cfb_chk_totalsecs ; 11/08/2023
00002747 807F0A30                       cmp     byte [bx+10], '0'
```

```
10047                                           ;cmp    byte [cs:bx+10], '0'
10048 0000274B 7519                             jne     short cfb_retit
10049
10050                                   cfb_chk_totalsecs:
10051                                           ; 11/08/2023
10052                                   ; 18/12/2023
10053                                   %if 0
10054                                           ; 17/10/2022
10055                                           mov     si, disksector+11 ; 14Eh+0Bh
10056                                           ;mov    si, 159h        ; disksector+EXT_BOOT.BPB
10057                                           ; 12/08/2023
10058                                           cmp     word [si+8], 0
10059                                           ;cmp    word [cs:si+8], 0 ; [cs:si+EBPB.TOTALSECTORS]
10060                                                                   ; just to make sure.
10061                                           jz      short cfb_retit
10062                                           ;mov    ax, [cs:si+8]  ; [cs:si+EBPB.TOTALSECTORS]
10063                                           ;add    ax, [cs:si+11h]        ; [cs:si+EBPB.HIDDENSECTORS]
10064                                           ; 12/08/2023
10065                                           mov     ax, [si+8]
10066                                           add     ax, [si+11h]
10067
10068                                           jnb     short cfb_retit
10069                                           jnz     short cfb_retit
10070                                                                   ; if carry set and ax=0
10071                                           dec     word [si+8]
10072                                           ;dec    word [cs:si+8] ; 0 -> 0FFFFh
10073                                                                   ; then decrease       BPB_TOTALSECTORS by 1
10074                                   %endif
10075                                           ; 18/12/2023
10076                                           ;cmp    word [bx+19], 0
10077 0000274D 8B4713                           mov     ax, [bx+19]    ; [bx+EBPB.TOTALSECTORS]
10078 00002750 21C0                             and     ax, ax ; 0 ?
10079 00002752 7412                             jz      short cfb_retit
10080
10081                                           ;mov    ax, [bx+19]
10082 00002754 03471C                           add     ax, [bx+28]    ; [bx+EBPB.HIDDENSECTORS]
10083 00002757 730D                             jnc     short cfb_retit
10084 00002759 750B                             jnz     short cfb_retit
10085                                           ; ax = 0                ; 0 -> 0FFFFh
10086 0000275B FF4F13                           dec     word [bx+19]   ; then decrease       BPB_TOTALSECTORS by 1
10087
10088 0000275E 836D1B01                         sub     word [di+1Bh], 1 ; [di+BDS.totalsecs32]
10089 00002762 835D1D00                         sbb     word [di+1Dh], 0 ; [di+BDS.totalsecs32+2]
10090                                   cfb_retit:
10091                                           ; 18/12/2023
10092                                           ;pop    si
10093                                           ;pop    dx
10094                                           ;pop    ax
10095
10096 00002766 C3                               retn
10097
10098                                   ; ---------------------------------------------------------------------------
10099
10100                                           ; 18/12/2023 - Retro DOS v5.0
10101                                           ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2A3Dh)
10102                                           ; ((MSDOS 6.22 IO.SYS - BIOSDATA:21DCh))
10103
10104 00002767 0200                     word2:          dw 2
10105 00002769 0300                     word3:          dw 3
10106 0000276B 0002                     word512:    dw 512
10107
10108                                   ; =============== S U B   R O U T I N E =======================================
10109
10110                                   ; 15/10/2022
10111
10112                                   ; setdrvparms sets up the recommended bpb in each bds in the system based on
10113                                   ; the form factor. it is assumed that the bpbs for the various form factors
10114                                   ; are present in the bpbtable. for hard files, the recommended bpb is the same
10115                                   ; as the bpb on the drive.
10116                                   ;
10117                                   ; no attempt is made to preserve registers since we are going to jump to
10118                                   ; sysinit straight after this routine.
10119
10120                                           ; 18/12/2023 - Retro DOS v5.0
10121                                           ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2A43h)
10122                                   setdrvparms:
10123                                           ; 12/12/2023
10124                                           ; ds = cs
10125 0000276D 31DB                             xor     bx, bx
10126                                           ; 18/10/2022
10127 0000276F C43E[1901]                       les     di, [start_bds] ; get first bds in list
10128                                   _next_bds:
10129 00002773 06                               push    es
10130 00002774 57                               push    di
10131
10132                                           ; 18/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
10133 00002775 268A5D3E                         mov     bl, [es:di+3Eh]        ; [es:di+BDS.formfactor]
10134                                           ;mov    bl, [es:di+22h]        ; [es:di+BDS.formfactor]
10135
10136 00002779 80FB05                           cmp     bl, 5          ; ffHardFile
10137 0000277C 753A                             jnz     short nothardff
10138 0000277E 31D2                             xor     dx, dx
10139 00002780 268B450E                         mov     ax, [es:di+0Eh]        ; [es:di+BDS.totalsecs16]
10140 00002784 09C0                             or      ax, ax
10141 00002786 7508                             jnz     short get_ccyl
10142 00002788 268B551D                         mov     dx, [es:di+1Dh]        ; [es:di+BDS.totalsecs32+2]
10143 0000278C 268B451B                         mov     ax, [es:di+1Bh]        ; [es:di+BDS.totalsecs32]
10144                                   get_ccyl:
10145 00002790 52                               push    dx
10146 00002791 50                               push    ax
10147 00002792 268B4515                         mov     ax, [es:di+15h]        ; [es:di+BDS.heads]
10148 00002796 26F76513                         mul     word [es:di+13h] ; [es:di+BDS.secpertrack]
10149                                                                   ; assume sectors per cyl. < 64k.
10150 0000279A 89C1                             mov     cx, ax         ; cx has # sectors per cylinder
10151 0000279C 58                               pop     ax
10152 0000279D 5A                               pop     dx             ; dx:ax = total sectors
10153 0000279E 50                               push    ax
10154 0000279F 89D0                             mov     ax, dx
10155 000027A1 31D2                             xor     dx, dx
10156 000027A3 F7F1                             div     cx
10157                                           ; 12/12/2023  ; !*!
10158                                           ; (data segment may not be same with code segment here)
10159                                           ;mov    [cs:temp_h], ax        ; ax be 0 here.
10160                                           ; 18/12/2023 - Retro DOS v5.0
10161                                           ;mov    [cs:saved_word], ax
10162 000027A5 58                               pop     ax
10163 000027A6 F7F1                             div     cx             ; div #sec by sec/cyl to get # cyl.
10164 000027A8 09D2                             or      dx, dx
10165 000027AA 7401                             jz      short no_cyl_rnd ; came out even
10166 000027AC 40                               inc     ax             ; round up
10167                                   no_cyl_rnd:
10168                                           ; 18/12/2023 - Retro DOS v5.0
10169 000027AD 26894541                         mov     [es:di+41h], ax        ; [es:di+BDS.cylinders]
10170                                           ;mov    [es:di+25h], ax        ; [es:di+BDS.cylinders]
```

```
10171
10172 000027B1 06                          push    es
10173 000027B2 1F                          pop     ds  ; !*! ; 12/12/2023
10174
10175 000027B3 8D7506                      lea     si, [di+6]     ; [di+BDS.bytespersec]
10176                                                             ; ds:si -> bpb for hard       file
10177 000027B6 EB55                        jmp     short set_recbpb
10178                                  ; ---------------------------------------------------------------------------
10179
10180                                  nothardff:
10181 000027B8 0E                          push    cs
10182 000027B9 1F                          pop     ds
10183
10184                                  ; if fake floppy drive variable is set then we don't have to handle this bds.
10185                                  ; we can just go and deal with the next bds at label go_to_next_bds.
10186
10187                                          ; 10/12/2022
10188                                          ; ds = cs
10189                                          ; 17/10/2022 (ds=cs)
10190 000027BA 803E[111A]01                    cmp     byte [fakefloppydrv], 1
10191                                          ;cmp    byte [cs:fakefloppydrv], 1
10192 000027BF 7454                          jz      short go_to_next_bds
10193 000027C1 80FB07                        cmp     bl, 7          ; ffother
10194                                                               ; special case "other" type of medium
10195 000027C4 753D                          jnz     short not_process_other
10196                                  process_other:
10197 000027C6 31D2                          xor     dx, dx
10198
10199                                          ;mov    ax, [di+25h]   ; [di+BDS.cylinders]
10200                                          ;mul    word [di+36h]  ; [di+BDS.rheads]
10201                                          ;mul    word [di+34h]  ; [di+BDS.rsecpertrack]
10202                                          ;mov    [di+2Fh], ax   ; [di+BDS.rtotalsecs16]
10203                                          ;                      ; have the total number of sectors
10204                                          ; 18/12/2023 - Retro DOS v5.0
10205 000027C8 8B4541                        mov     ax, [di+41h]   ; [di+BDS.cylinders]
10206 000027CB F76552                        mul     word [di+52h]  ; [di+BDS.rheads]
10207 000027CE F76550                        mul     word [di+50h]  ; [di+BDS.rsecpertrack]
10208 000027D1 89454B                        mov     [di+4Bh], ax   ; [di+BDS.rtotalsecs16]
10209                                                               ; have the total number of sectors
10210 000027D4 48                            dec     ax
10211 000027D5 B201                          mov     dl, 1
10212                                  _again:
10213 000027D7 3DF60F                        cmp     ax, 0FF6h      ; 4096-10
10214 000027DA 7206                          jb      short _@@
10215 000027DC D1E8                          shr     ax, 1
10216 000027DE D0E2                          shl     dl, 1
10217 000027E0 EBF5                          jmp     short _again
10218                                  ; ---------------------------------------------------------------------------
10219
10220                                  _@@:
10221 000027E2 80FA01                        cmp     dl, 1          ; is it a small disk ?
10222 000027E5 7405                          jz      short __@@     ; yes, 224 root       entries is enuf
10223
10224                                          ; 18/12/2023 - Retro DOS v5.0
10225 000027E7 C74549F000                    mov     word [di+49h], 240 ; [di+BDS.rdirentries]
10226                                          ;mov    word [di+2Dh], 240 ; [di+BDS.rdirentries]
10227                                  __@@:
10228                                          ; 18/12/2023 - Retro DOS v5.0
10229 000027EC 885545                        mov     [di+45h], dl   ; [di+BDS.rsecperclus]
10230                                          ;mov    [di+29h], dl   ; [di+BDS.rsecperclus]
10231
10232                                  ; logic to get the sectors/fat area.
10233                                  ; fat entry is assumed to be 1.5 bytes!!!
10234
10235                                          ; 10/12/2022
10236                                          ; ds = cs
10237                                          ; 17/10/2022 (ds=cs)
10238 000027EF F726[6927]                      mul     word [word3]   ; * 3
10239 000027F3 F736[6727]                      div     word [word2]   ; / 2
10240 000027F7 31D2                          xor     dx, dx
10241 000027F9 F736[6B27]                    div     word [word512] ; / 512
10242                                                               ;
10243                                          ; 10/12/2022
10244                                          ;mul    word [cs:word3]        ; * 3
10245                                          ;div    word [cs:word2]        ; / 2
10246                                          ;xor    dx, dx
10247                                          ;div    word [cs:word512] ; / 512
10248                                                               ;
10249 000027FD 40                            inc     ax             ; + 1
10250                                  no_round_up:
10251                                          ; 18/12/2023 - Retro DOS v5.0
10252 000027FE 89454E                        mov     [di+4Eh], ax   ; [di+BDS.rfatsecs]
10253                                          ;mov    [di+32h], ax   ; [di+BDS.rfatsecs]
10254
10255 00002801 EB12                          jmp     short go_to_next_bds
10256                                  ; ---------------------------------------------------------------------------
10257
10258                                  not_process_other:
10259 00002803 D1E3                          shl     bx, 1          ; bx is word index into       table of bpbs
10260
10261                                          ;mov    si, bpbtable
10262                                          ;mov    si, [bpbtable+bx] ; 15/10/2022
10263                                          ; 09/12/2022
10264                                          ;mov    si, BPBTABLE
10265                                          ;mov    si, [bx+si]    ; get address of bpb
10266                                          ; 10/12/2022
10267                                          ;mov    si, [BPBTABLE+bx]
10268                                          ; 13/12/2022
10269                                          ;mov    si, [SYSINITOFFSET+bpbtable+bx] ; wrong ! 14/08/2023
10270
10271                                          ; 14/08/2023
10272                                          SYSINIT_OFFSET equ (SYSINITSEG-DOSBIODATASEG<<4)
10273                                                                        ; correct offset
10274 00002805 8BB7[6E99]                      mov     si, [bx+SYSINIT_OFFSET+bpbtable]
10275
10276                                          ; 18/12/2023
10277                                          ; si = address of the requested disk(ette) parameter block
10278                                          ;      ! as offset from SYSINIT segment !
10279
10280                                          ; 28/08/2023
10281 00002809 81C64049                      add     si, SYSINIT_OFFSET
10282                                                 ; + displacement from BIOSDATA segment ; 18/12/2023
10283                                  set_recbpb:
10284                                          ; 18/12/2023
10285                                          ;lea    di, [di+27h]   ; [di+BDS.R_BPB]
10286                                          ;                      ; es:di -> recbpb
10287                                          ;mov    cx, 25         ; bpbx.size
10288                                          ;rep movsb             ; move (size bpbx) bytes
10289
10290                                          ; 18/12/2023 - Retro DOS v5.0
10291 0000280D 8D7D43                        lea     di, [di+43h]   ; [di+BDS.R_BPB]
10292                                                               ; es:di -> recbpb
10293 00002810 B93500                        mov     cx, 53         ; bpbx.size
10294 00002813 F3A4                          rep movsb              ; move (size bpbx) byte
```

```
10295                                   go_to_next_bds:
10296 00002815 5F                                 pop     di
10297 00002816 07                                 pop     es              ; restore pointer to bds
10298 00002817 26C43D                             les     di, [es:di]     ; [es:di+BDS.link]
10299 0000281A 83FFFF                             cmp     di, 0FFFFh      ; -1
10300 0000281D 740A                               jz      short got_end_of_bds_chain
10301 0000281F E951FF                             jmp     _next_bds
10302
10303                                   ; ---------------------------------------------------------------------------
10304
10305                                             ; 18/12/2022
10306                                   ;got_end_of_bds_chain:
10307                                             ;retn
10308
10309                                   ; =============== S U B   R O U T I N E =======================================
10310
10311                                   ; 15/10/2022
10312                                   ; 30/12/2018 - Retro DOS v4.0
10313
10314                                   ; al = device number
10315
10316                                   print_init:
10317 00002822 98                                 cbw
10318 00002823 89C2                               mov     dx, ax
10319 00002825 B401                               mov     ah, 1
10320 00002827 CD17                               int     17h             ; PRINTER - INITIALIZE
10321                                                                     ; DX = printer port (0-3)
10322                                                                     ; Return: AH = status
10323                                   got_end_of_bds_chain:     ; 18/12/2022
10324 00002829 C3                                 retn
10325
10326                                   ; =============== S U B   R O U T I N E =======================================
10327
10328                                   ; al = device number
10329
10330                                   aux_init:
10331 0000282A 98                                 cbw
10332 0000282B 89C2                               mov     dx, ax
10333                                             ;mov    al, 0A3h        ; RSINIT ; 0A3h
10334                                                                     ; 2400,n,1,8 (msequ.inc)
10335                                             ;mov    ah, 0
10336                                             ; 10/12/2022
10337 0000282D B8A300                             mov     ax, 00A3h
10338 00002830 CD14                               int     14h             ; SERIAL I/O - INITIALIZE USART
10339                                                                     ;       AL = initializing parameters,
10340                                                                     ;       DX = port number (0-3)
10341                                                                     ; Return: AH = RS-232 status code bits,
10342                                                                     ;         AL = modem status bits
10343 00002832 C3                                 retn
10344
10345                                 ; =============== S U B   R O U T I N E =======================================
10346
10347                                 ; 18/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
10348                                 ; 08/08/2023 - Retro DOS v4.2 (Modified MSDOS 6.22 IO.SYS)
10349                                 ; 15/10/2022 (Modified MSDOS 5.0 IO.SYS) -Retro DOS v4 2022- (MSDOS 5.0-6.21)
10350                                 ; 30/12/2018 - Retro DOS v4.0
10351                                 ; 03/06/2018 - Retro DOS v3.0
10352                                 ; (19/03/2018 - Retro DOS v2.0)
10353
10354                                 ; domini ******************************************************************************
10355                                 ;
10356                                 ;mini disk initialization routine. called right after dohard
10357                                 ;modified for >2 hardfile support
10358                                 ;
10359                                 ; **cs=ds=es=datagrp
10360                                 ;
10361                                 ; **domini will search for every extended partition in the system, and
10362                                 ;   initialize it.
10363                                 ;
10364                                 ; **bdsm stands for bds table for mini disk and located right after the label
10365                                 ;   end96tpi. end_of_bdsm will have the offset value of the ending
10366                                 ;   address of bdsm table.
10367                                 ;
10368                                 ; **bdsm is the same as usual bds structure except that tim_lo, tim_hi entries
10369                                 ;   are overlapped and used to identify mini disk and the number of hidden_trks.
10370                                 ;   right now, they are called as ismini, hidden_trks respectively.
10371                                 ;
10372                                 ; **domini will use the same routine in sethard routine after label set2 to
10373                                 ;   save coding.
10374                                 ;
10375                                 ; **drvmax determined in dohard routine will be used for the next
10376                                 ;   available logical mini disk drive number.
10377                                 ;
10378                                 ; input: drvmax, dskdrvs
10379                                 ;
10380                                 ; output: minidisk installed. bdsm table established and installed to bds.
10381                                 ;     end_of_bdsm - ending offset address of bdsm.
10382                                 ;
10383                                 ; called modules:
10384                                 ;             getboot
10385                                 ;             find_mini_partition (new), xinstall_bds (new), M038
10386                                 ;
10387                                 ;             setmini (new, it will use set2 routine)
10388                                 ;
10389                                 ; variables used: end_of_bdsm
10390                                 ;             rom_minidisk_num
10391                                 ;             mini_hdlim, mini_seclim
10392                                 ;             BDS_STRUC, start_bds
10393                                 ;
10394                                 ;*******************************************************************************
10395
10396                                             ; 18/12/2023 - Retro DOS v5.0 IO.SYS/IBMBIO.COM
10397                                             ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2B10h)
10398
10399                                             ; 19/10/2022
10400                                 domini:
10401 00002833 8A36[5D1A]                       mov     dh, [hnum]      ; get number of hardfiles
10402                                             ; 10/12/2022
10403 00002837 20F6                             and     dh, dh
10404                                           ;cmp    dh, 0
10405 00002839 743C                             jz      short dominiret       ; no hard file?       then exit.
10406 0000283B B280                             mov     dl, 80h         ; start with hardfile 80h
10407                                 domini_loop:
10408                                             ; 18/12/2023 - Retro DOS v5.0
10409 0000283D 31C0                             xor     ax, ax ; 0
10410                                             ; ds = cs
10411                                             ;mov    [cs:ep_start_sector], ax
10412                                             ;mov    [cs:ep_start_sector+2], ax
10413                                             ;mov    [cs:ep_hidden_secs], ax
10414                                             ;mov    [cs:ep_hidden_secs+2], ax
10415 0000283F A3[8323]                          mov     [ep_start_sector], ax
10416 00002842 A3[8523]                         mov     [ep_start_sector+2], ax
10417 00002845 A3[8723]                         mov     [ep_hidden_secs], ax
10418 00002848 A3[8923]                         mov     [ep_hidden_secs+2], ax
```

```
10419                                   ;
10420 0000284B 52                       push    dx
10421 0000284C 8816[5C1A]               mov     [rom_minidisk_num], dl
10422 00002850 B408                     mov     ah, 8
10423 00002852 CD13                     int     13h             ; DISK - DISK -       GET CURRENT DRIVE PARAMETERS (XT,AT,XT286,CONV,PS)
10424                                                           ; DL = drive number
10425                                                           ; Return: CF set on error, AH =      status code, BL       = drive type
10426                                                           ; DL = number of consecutive drives
10427                                                           ; DH = maximum value for head number, ES:DI -> drive parameter
10428
10429                                   ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
10430                                   ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:2B36h
10431                                   ;inc    dh
10432                                   ;xor    ax, ax
10433                                   ;mov    al, dh
10434 00002854 31C0                     xor     ax, ax
10435 00002856 88F0                     mov     al, dh  ; <= 255
10436 00002858 40                       inc     ax      ; (0FFh -> 100h)
10437 00002859 A3[621A]                 mov     [mini_hdlim], ax ; # of heads
10438                                   ;and    cl, 3Fh
10439                                   ;mov    al, cl
10440                                   ; 08/08/2023
10441 0000285C 88C8                     mov     al, cl
10442 0000285E 83E03F                   and     ax, 3Fh
10443 00002861 A3[641A]                 mov     [mini_seclim], ax ; # of sectors/track
10444
10445                                   ; 18/12/2023
10446                                   ;push   es ; * ; not necessary
10447 00002864 8A16[5C1A]               mov     dl, [rom_minidisk_num]
10448 00002868 E860FA                   call    getboot         ; read master boot record into
10449                                                           ; initbootsegment:bootbias
10450 0000286B 7203                     jc      short domininext
10451 0000286D E80800                   call    find_mini_partition
10452                                 domininext:
10453                                   ;pop    es ; *
10454 00002870 5A                       pop     dx
10455 00002871 FEC2                     inc     dl              ; next hard file
10456 00002873 FECE                     dec     dh
10457 00002875 75C6                     jnz     short domini_loop
10458                                 dominiret:
10459 00002877 C3                       retn
10460
10461                                 ; =============== S U B   R O U T I N E =======================================
10462
10463                                 ; 15/10/2022 (Modified MSDOS 5.0 IO.SYS)
10464                                 ; 30/12/2018 - Retro DOS v4.0
10465
10466                                 ;find_mini_partition tries to find every extended partition on a disk.
10467                                 ;at entry:  di -> bdsm entry
10468                                 ;           es:bx -> 07c0:bootbias - master boot record
10469                                 ;           rom_minidisk_num - rom drive number
10470                                 ;           drvmax - logical drive number
10471                                 ;           mini_hdlim, mini_seclim
10472                                 ;
10473                                 ;called routine: setmini which uses set2 (in sethard routine)
10474                                 ;variables & equates used from original bios - flags, fnon_removable, fbigfat
10475
10476                                           ; 19/12/2023 - Retro DOS v5.0
10477                                           ;       (Modified PCDOS 7.1 IBMBIO.COM)
10478                                           ; (PCDOS 7.1 IBMBIO.COM - BIOSADATA:2BFCh)
10479
10480                                 find_mini_partition:
10481 00002878 81C3C201                       add     bx, 1C2h        ; bx -> file system id
10482
10483                                           ; 19/12/2023
10484                                           ; PCDOS 7.1 IBMBIO.COM
10485                                           ;mov    word [ld_p_number], 26
10486                                 fmpnext:
10487                                           ;add    word [ld_p_number], 16
10488                                           ;cmp    word [ld_p_number], 4122
10489                                           ;               ; 64 logical disk partitions (64 EBRs)
10490                                           ;               ; (64*4 = 256 pte's, 256*16 = 4096, + 26 = 4122)
10491                                           ;jg     short fmpnextfound
10492
10493 0000287C 26803F05                       cmp     byte [es:bx], 5 ; 05h = extended partition id.
10494 00002880 7410                           je      short fmpgot ; Extended DOS CHS
10495
10496                                           ; 19/12/2023 - Retro DOS v5.0
10497 00002882 26803F0F                       cmp     byte [es:bx], 0Fh ; Extended DOS LBA
10498 00002886 740A                           je      short fmpgot
10499
10500 00002888 83C310                         add     bx, 16
10501 0000288B 81FB0204                       cmp     bx, 402h        ; 202h+bootbias
10502 0000288F 75EB                           jne     short fmpnext
10503                                         ;jmp    short fmpnextfound ; extended partition     not found
10504                                         ; 18/12/2022
10505                                 fmpnextfound:
10506 00002891 C3                             retn
10507
10508                                 ;               ; 30/07/2019 - Retro DOS v3.2
10509                                 ;               jb      short fmpnext
10510                                 ;fmpret:
10511                                 ;               retn    ; 29/05/2019
10512
10513                                 ; ---------------------------------------------------------------------------
10514
10515                                           ; 19/10/2022
10516                                 fmpgot:                                 ; found my partition.
10517 00002892 E82B01                         call    dmax_check      ; check for drvmax already 26
10518 00002895 73FA                           jnb     short fmpnextfound ; done if too many
10519
10520 00002897 8B3E[601A]                     mov     di, [end_of_bdss] ; get next free bds
10521
10522                                         ; 19/12/2023
10523                                         ;mov    word [di+47h], 1 ; [di+BDS.bdsm_ismini]
10524                                         ;; 10/12/2022
10525                                         ;or     byte [di+23h], 1
10526                                         ;;or    word [di+23h], 1 ; [di+BDS.flags]
10527                                         ;                       ; fNon_Removable
10528                                         ;mov    byte [di+22h], 5 ; [di+BDS.formfactor]
10529                                         ;                       ; ffHardFile
10530                                         ; 19/12/2023 - Retro DOS v5.0
10531 0000289B C745790100                     mov     word [di+79h], 1 ; [di+BDS.bdsm_ismini]
10532 000028A0 804D3F01                       or      byte [di+3Fh], 1 ; [di+BDS.flags], fNon_Removable
10533 000028A4 C6453E05                       mov     byte [di+3Eh], 5 ; [di+BDS.formfactor], ffHardFile
10534
10535 000028A8 C606[061A]00                   mov     byte [fbigfat], 0 ; assume 12 bit fat.
10536 000028AD A1[621A]                       mov     ax, [mini_hdlim]
10537 000028B0 894515                         mov     [di+15h], ax    ; [di+BDS.heads]
10538 000028B3 A1[641A]                       mov     ax, [mini_seclim]
10539 000028B6 894513                         mov     [di+13h], ax    ; [di+BDS.secpertrack]
10540 000028B9 A0[5C1A]                       mov     al, [rom_minidisk_num]
10541 000028BC 884504                         mov     [di+4], al      ; [di+BDS.drivenum]
10542                                                                 ; set physical number
```

```
10543 000028BF A0[7500]                           mov     al, [drvmax]
10544 000028C2 884505                             mov     [di+5],al       ; [di+BDS.drivelet]
10545                                                                     ; set logical number
10546 000028C5 26837F0A00                         cmp     word [es:bx+10], 0
10547                                             ;ja     short fmpgot_cont
10548 000028CA 7707                               ja      short fmpgot1  ; 19/12/2023
10549 000028CC 26837F0840                         cmp     word [es:bx+8], 64 ; with current bpb,
10550                                                                     ; only lower word is meaningful.
10551 000028D1 72BE                               jb      short fmpnextfound
10552                                                                     ; should be bigger than 64 sectors at least
10553                                     fmpgot1:    ; 19/12/2023
10554                                     ;fmpgot_cont:
10555 000028D3 83EB04                             sub     bx, 4           ; let bx point to the start of the entry
10556 000028D6 268A7702                           mov     dh, [es:bx+2]  ; cylinder
10557 000028DA 80E6C0                             and     dh, 0C0h        ; get higher bits of cyl
10558 000028DD D0C6                               rol     dh, 1
10559 000028DF D0C6                               rol     dh, 1
10560 000028E1 268A5703                           mov     dl, [es:bx+3]  ; cyl byte
10561                                             ; 19/12/2023 - Retro DOS v5.0
10562 000028E5 89557B                             mov     [di+7Bh], dx   ; [di+BDS.bdsm_hidden_trks]
10563                                             ;mov    [di+49h], dx   ; [di+BDS.bdsm_hidden_trks]
10564                                                                     ; set hidden trks
10565                                             ; 19/12/2023
10566                                             ;push   bx ; * ; PCDOS 7.1
10567                                             ;;;
10568 000028E8 268B4F08                           mov     cx, [es:bx+8]  ; partition size, lw
10569 000028EC 268B470A                           mov     ax, [es:bx+10] ; partition size, hw
10570 000028F0 030E[8323]                         add     cx, [ep_start_sector]
10571 000028F4 1306[8523]                         adc     ax, [ep_start_sector+2]
10572 000028F8 31D2                               xor     dx, dx ; 19/12/2023
10573 000028FA 3916[8323]                         cmp     [ep_start_sector], dx ; 0
10574                                             ;cmp    word [ep_start_sector], 0
10575 000028FE 750D                               jnz     short fmpgot2
10576 00002900 3916[8523]                         cmp     [ep_start_sector+2], dx ; 0
10577                                             ;cmp    word [ep_start_sector+2], 0
10578 00002904 7507                               jnz     short fmpgot2
10579 00002906 890E[8323]                         mov     [ep_start_sector], cx
10580 0000290A A3[8523]                           mov     [ep_start_sector+2], ax
10581                                     fmpgot2:
10582 0000290D 890E[8723]                         mov     [ep_hidden_secs], cx
10583 00002911 A3[8923]                           mov     [ep_hidden_secs+2], ax
10584
10585                                             ; convert start sector address to CHS
10586
10587                                             ; 19/12/2023
10588                                             ; dx = 0
10589                                             ;push   bx ; * ; not necessary
10590
10591                                             ;mov    bx, [di+13h]   ; [di+BDS.secpertrack]
10592 00002914 8B7513                             mov     si, [di+13h]   ; [di+BDS.secpertrack]
10593                                             ;xor    dx, dx  ; dx = 0
10594                                             ;div    bx
10595 00002917 F7F6                               div     si
10596 00002919 91                                 xchg    ax, cx
10597                                             ;div    bx
10598 0000291A F7F6                               div     si
10599                                             ;mov    bx, [di+15h]   ; [di+BDS.heads]
10600                                             ; 17/04/2024 (BugFix)
10601 0000291C 8B7515                             mov     si, [di+15h]   ; [di+BDS.heads]
10602 0000291F 91                                 xchg    ax, cx
10603 00002920 31D2                               xor     dx, dx
10604                                             ;div    bx
10605 00002922 F7F6                               div     si
10606 00002924 91                                 xchg    ax, cx
10607                                             ;div    bx
10608 00002925 F7F6                               div     si
10609
10610                                             ;pop    bx ; *
10611
10612 00002927 09C9                               or      cx, cx
10613 00002929 7505                               jnz     short fmpgot_lba_rd
10614 0000292B 3D0004                             cmp     ax, 1024        ; cylinder number < 1024, CHS read is proper
10615 0000292E 7235                               jb      short fmpgot_chs_rd
10616                                     fmpgot_lba_rd:
10617 00002930 804D4004                           or      byte [di+40h], 4 ; set fLBArw flag ; LBA read/write ok/ready
10618 00002934 8A16[5C1A]                         mov     dl, [rom_minidisk_num]
10619 00002938 1E                                 push    ds
10620                                             ; 19/12/2023
10621                                             ;push   si ; ** ; not necessary
10622 00002939 31C0                               xor     ax, ax          ; push bp
10623                                                                      ; mov bp, sp ; (*)
10624 0000293B 50                                 push    ax ; 0
10625 0000293C 50                                 push    ax ; 0
10626 0000293D FF36[8923]                         push    word [ep_hidden_secs+2]
10627 00002941 FF36[8723]                         push    word [ep_hidden_secs]
10628 00002945 B80002                             mov     ax, bootbias ; 200h
10629                                             ;mov    ax, 200h        ; bootbias (buffer offset)
10630 00002948 06                                 push    es              ; buffer segment
10631 00002949 50                                 push    ax
10632 0000294A B80100                             mov     ax, 1
10633 0000294D 50                                 push    ax              ; read count
10634 0000294E B81000                             mov     ax, 10h         ; DAP size = 16
10635 00002951 50                                 push    ax
10636 00002952 8CD0                               mov     ax, ss
10637 00002954 8ED8                               mov     ds, ax
10638 00002956 89E6                               mov     si, sp          ; ds:si = Disk Address Packet
10639
10640 00002958 B442                               mov     ah, 42h         ; LBA read
10641 0000295A CD13                               int     13h             ; DISK - IBM/MS Extension
10642                                                                     ; EXTENDED READ (DL - drive, DS:SI - disk address packet)
10643                                             ; 19/12/2023
10644                                             ;pushf          ; PCDOS 7.1 IBMBIO.COM BUG! Erdogan Tan - 08/08/2023
10645                                             ;add    sp, 16
10646                                             ;popf           ; BUG!
10647                                                                     ; mov sp, bp ; (*)
10648                                                                     ; pop bp
10649                                             ; 19/12/2023
10650 0000295C 9F                                 lahf            ; load status flags into AH
10651 0000295D 83C410                             add     sp, 16
10652 00002960 9E                                 sahf            ; store AH into flags
10653
10654                                             ;pop    si ; ** ; 19/12/2023
10655 00002961 1F                                 pop     ds
10656 00002962 7317                               jnc     short fmpgot3
10657                                     fmpnotfound:        ; 19/12/2023
10658 00002964 C3                                 retn
10659                                             ;jmp    short fmpgot3
10660                                             ;;;
10661
10662                                             ; 19/12/2023
10663                                     fmpgot_chs_rd:
10664 00002965 268B4F02                           mov     cx, [es:bx+2]  ; cylinder,cylinder/sector
10665 00002969 268A7701                           mov     dh, [es:bx+1]  ; head
10666 0000296D 8A16[5C1A]                         mov     dl, [rom_minidisk_num]
```

```
10667 00002971 BB0002                          mov     bx, 200h        ; bootbias
10668 00002974 B80102                          mov     ax, 201h
10669 00002977 CD13                            int     13h             ; DISK - READ SECTORS INTO MEMORY
10670                                                                  ; AL = number of sectors to read, CH = track, CL = sector
10671                                                                  ; DH = head, DL       = drive, ES:BX -> buffer to fill
10672                                                                  ; Return: CF set on error, AH =      status, AL = number of sectors read
10673                                  ;fmpgot3:      ; 19/12/2023
10674                                                 ;jc     short fmpnextfound
10675 00002979 72E9                              jc      short fmpnotfound
10676                                  fmpgot3:
10677 0000297B BBC203                            mov     bx, 3C2h        ; 1C2h+bootbias
10678
10679                                            ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
10680                                            ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:2C7Ch
10681 0000297E 26817F3C55AA                      cmp     word [es:bx+3Ch], 0AA55h ; 03C2h+03Ch = 3FEh
10682                                            ;jne    short fmpnextfound ; not a valid boot sector !
10683                                            ; 19/12/2023
10684 00002984 75DE                              jne     short fmpnotfound ; not a valid boot sector !
10685
10686                                            ; 13/08/2023
10687                                            ;push   es
10688 00002986 E80800                            call    setmini         ; install a mini disk.
10689                                                                    ; bx value saved.
10690                                            ;pop    es  ; 13/08/2023
10691 00002989 7203                              jc      short fmpnextchain
10692 0000298B E84700                            call    xinstall_bds    ; -- install the bdsm into table
10693                                  fmpnextchain:
10694 0000298E E9EBFE                            jmp     fmpnext         ; let's find out
10695                                                                    ; if we have any chained partition
10696                                  ; ---------------------------------------------------------------------------
10697
10698                                            ; 18/12/2022
10699                                  ;fmpnextfound:
10700                                            ;retn
10701
10702                                  ; =============== S U B   R O U T I N E =======================================
10703
10704                                  ; 15/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
10705                                  ; 28/12/2018 - Retro DOS v4.0 (MSDOS 6.21)
10706
10707                                  ; 19/12/2022 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
10708                                  ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2C92h)
10709
10710                                  setmini:    ; 'setmini' is called from 'find_mini_partition' procedure
10711
10712 00002991 57                              push    di
10713 00002992 53                              push    bx
10714                                          ; 12/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
10715                                          ; ds = cs = BIOSDATA segment
10716                                          ;push   ds
10717 00002993 06                              push    es
10718                                  setmini_1:
10719                                          ;cmp    byte [es:bx], 1 ; FAT12 partition
10720                                          ;je     short setmini_2
10721                                          ;cmp    byte [es:bx], 4 ; FAT16 (CHS) partition
10722                                          ;je     short setmini_2
10723                                          ;cmp    byte [es:bx], 6 ; FAT16 BIG (CHS) partition
10724                                          ;je     short setmini_2
10725                                          ;
10726                                          ; 19/12/2023 - Retro DOS v5.0
10727                                          ;cmp    byte [es:bx], 0Bh ; FAT32 (CHS) partition
10728                                          ;je     short setmini_2
10729                                          ;cmp    byte [es:bx], 0Ch ; FAT32 (LBA) partition
10730                                          ;je     short setmini_2
10731                                          ;cmp    byte [es:bx], 0Eh ; FAT16 (LBA) partition
10732                                          ;je     short setmini_2
10733
10734                                          ; 19/12/2023
10735 00002994 268A07                          mov     al, [es:bx]
10736 00002997 3C01                            cmp     al, 1           ; FAT12 partition
10737 00002999 7422                            je      short setmini_2
10738 0000299B 3C04                            cmp     al, 4           ; FAT16 (CHS) partition
10739 0000299D 741E                            je      short setmini_2
10740 0000299F 3C06                            cmp     al, 6           ; FAT16 BIG (CHS) partition
10741 000029A1 741A                            je      short setmini_2
10742 000029A3 3C0B                            cmp     al, 0Bh         ; FAT32 (CHS) partition
10743 000029A5 7416                            je      short setmini_2
10744 000029A7 3C0C                            cmp     al, 0Ch         ; FAT32 (LBA) partition
10745 000029A9 7412                            je      short setmini_2
10746 000029AB 3C0E                            cmp     al, 0Eh         ; FAT16 (LBA) partition
10747 000029AD 740E                            je      short setmini_2
10748
10749 000029AF 83C310                          add     bx, 16
10750 000029B2 81FB0204                        cmp     bx, 402h        ; 202h+bootbias
10751                                          ;jne    short setmini_1
10752 000029B6 72DC                            jb      short setmini_1 ; 19/12/2023
10753 000029B8 F9                              stc
10754 000029B9 07                              pop     es
10755                                          ; 12/08/2023
10756                                          ;pop    ds
10757 000029BA 5B                              pop     bx
10758 000029BB 5F                              pop     di
10759 000029BC C3                              retn
10760
10761                                  ; ---------------------------------------------------------------------------
10762                                  setmini_2:
10763 000029BD E9CBF9                          jmp     set2            ; branch into middle of sethard
10764
10765                                  ; =============== S U B   R O U T I N E =======================================
10766
10767                                  ; 30/12/2022 - Retro DOS v4.2
10768                                  ; (SYSINITSEG is 473h for MSDOS 6.21 IO.SYS)
10769
10770                                  ; 15/10/2022
10771                                  ; 28/12/2018 - Retro DOS v4.0
10772                                  ;
10773                                  ; dmax_check -- call this when we want to install a new drive.
10774                                  ;           it checks for drvmax < 26 to see if there is
10775                                  ;           a drive letter left.
10776                                  ;
10777                                  ;   drvmax < 26 : carry SET!
10778                                  ;   drvmax >=26 : carry RESET!, error flag set for message later
10779                                  ;                 trash ax
10780
10781                                          ; 19/12/2023 - Retro DOS v5.0
10782                                  dmax_check:
10783 000029C0 803E[7500]1A                    cmp     byte [drvmax], 26 ; checks for drvmax < 26
10784 000029C5 720D                            jb      short dmax_ok  ; return with carry if okay
10785 000029C7 06                              push    es
10786                                          ;;mov   ax, 46Dh        ; SYSINIT_SEG (SYSINIT segment)
10787                                          ;mov    ax, 544h        ; 19/12/2023 (PCDOS 7.1)
10788 000029C8 B80405                          mov     ax, SYSINITSEG ; 17/10/2022
10789 000029CB 8EC0                            mov     es, ax
10790                                          ; 18/10/2022
```

```
10791 000029CD 26C606[8803]01                   mov     byte [es:TOOMANYDRIVESFLAG], 1 ; 09/12/2022
10792                                           ;mov    byte ptr es:3FFh, 1 ; [es:toomanydrivesflag]
10793                                                                       ; set message flag
10794                                                                       ; [SYSINIT+toomanydrivesflag]
10795 000029D3 07                               pop     es
10796
10797                                           ;;push  es
10798                                           ;;mov   ax,SYSINIT_SEG
10799                                           ;;mov   es,ax
10800                                           ;;mov   byte [es:toomanydrivesflag],1
10801                                                                       ; set message flag
10802                                           ;;pop   es
10803                                           ;
10804                                           ;mov    byte [SYSINIT+toomanydrivesflag],1
10805                                   dmax_ok:
10806 000029D4 C3                               retn
10807
10808                                   ; =============== S U B   R O U T I N E =======================================
10809
10810                                   ; 18/10/2022
10811                                   ; 15/10/2022
10812                                   ; 28/12/2018 - Retro DOS v4.0
10813                                   ;
10814                                   ;   link next bds (at ds:di) into the chain. assume that the
10815                                   ;     chain is entirely within ds == datagrp. also update drvmax,
10816                                   ;     dskdrv_table, and end_of_bdss.
10817
10818                                           ; 19/12/2023 - Retro DOS v5.0
10819                                           ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2CE1h)
10820                                   xinstall_bds:
10821 000029D5 56                               push    si
10822 000029D6 53                               push    bx
10823 000029D7 8B36[1901]                       mov     si, [start_bds]       ; get first bds
10824                                   xinstall_bds_1:
10825 000029DB 833CFF                           cmp     word [si], 0FFFFh ; is this the last one?
10826 000029DE 7404                             jz      short xinstall_bds_2 ;skip ahead if so
10827                                           ;mov    si, [si+BDS.link]
10828 000029E0 8B34                             mov     si, [si]       ; chain through list
10829 000029E2 EBF7                             jmp     short xinstall_bds_1
10830
10831                                   xinstall_bds_2:
10832                                           ;mov    [si+BDS.link], di
10833 000029E4 893C                             mov     [si], di
10834                                           ;mov    [si+BDS.link+2], ds
10835 000029E6 8C5C02                           mov     [si+2], ds
10836                                           ;mov    word [di+BDS.link], -1
10837 000029E9 C705FFFF                         mov     word [di], 0FFFFh ; make sure it is a null ptr.
10838                                           ;mov    [di+BDS.link+2], ds
10839 000029ED 8C5D02                           mov     [di+2], ds ; might as well plug segment
10840                                           ; 20/03/2019 - Retro DOS v4.0
10841                                           ;lea    bx, [di+BDS.BPB]
10842 000029F0 8D5D06                           lea     bx, [di+6]
10843 000029F3 8B36[5E1A]                       mov     si, [last_dskdrv_table]
10844 000029F7 891C                             mov     [si], bx
10845 000029F9 8306[5E1A]02                     add     word [last_dskdrv_table], 2
10846 000029FE FE06[7500]                       inc     byte [drvmax]
10847                                           ;add    word [end_of_bdss], 100 ; BDS.size = 100
10848                                           ; 19/12/2023 - Retro DOS v5.0
10849 00002A02 8106[601A]9600                   add     word [end_of_bdss], 150 ; BDS.size = 150
10850 00002A08 5B                               pop     bx
10851 00002A09 5E                               pop     si
10852 00002A0A C3                               retn
10853
10854                                   ; =============== S U B   R O U T I N E =======================================
10855
10856                                   ; 17/10/2022
10857                                   ; 15/10/2022
10858                                   ; 28/12/2018 - Retro DOS v4.0
10859                                   ; 03/06/2018 - Retro DOS v3.0
10860
10861                                           ; 19/12/2023 - Retro DOS v5.0
10862                                   cmos_clock_read:
10863 00002A0B 50                               push    ax
10864 00002A0C 51                               push    cx
10865 00002A0D 52                               push    dx
10866 00002A0E 55                               push    bp
10867 00002A0F 31ED                             xor     bp, bp
10868                                   loop_clock:
10869 00002A11 31C9                             xor     cx, cx
10870 00002A13 31D2                             xor     dx, dx
10871 00002A15 B402                             mov     ah, 2
10872 00002A17 CD1A                             int     1Ah             ; CLOCK - READ REAL TIME CLOCK (AT,XT286,CONV,PS)
10873                                                                   ; Return: CH = hours in       BCD
10874                                                                   ; CL = minutes in BCD
10875                                                                   ; DH = seconds in BCD
10876                                           ; 19/12/2023
10877                                           ;cmp    cx, 0
10878 00002A19 21C9                             and     cx, cx
10879 00002A1B 750F                             jnz     short clock_present
10880                                           ;cmp    dx, 0
10881 00002A1D 09D2                             or      dx, dx
10882 00002A1F 750B                             jnz     short clock_present
10883                                           ;cmp    bp, 1           ; read again after a slight delay, in case clock
10884                                           ;je     short no_readdate ; was       at zero setting.
10885 00002A21 21ED                             and     bp, bp
10886 00002A23 751A                             jnz     short no_readdate
10887 00002A25 45                               inc     bp              ; only perform delay once.
10888                                           ;mov    cx, 4000h       ; 16384
10889                                           ; 19/12/2023
10890 00002A26 B540                             mov     ch, 40h ; cx = 4000h ; 16384
10891                                   delay:
10892 00002A28 E2FE                             loop    delay
10893 00002A2A EBE5                             jmp     short loop_clock
10894                                   ; ---------------------------------------------------------------------------
10895
10896                                   clock_present:
10897                                           ;mov    byte [cs:havecmosclock], 1 ; set the flag for cmos clock
10898                                           ; 19/12/2023
10899                                           ; ds = cs
10900 00002A2C C606[8C04]01                     mov     byte [havecmosclock], 1 ; set the flag for cmos clock
10901
10902 00002A31 E81000                           call    cmosck          ; reset cmos clock rate       that may be
10903                                                                   ; possibly destroyed by       cp dos and
10904                                                                   ; post routine did not restore that.
10905 00002A34 56                               push    si
10906 00002A35 E833EE                           call    read_real_date ; read real-time clock for date
10907 00002A38 FA                               cli
10908                                           ;mov    ds:daycnt, si  ; set system date
10909 00002A39 8936[8904]                       mov     [daycnt], si
10910 00002A3D FB                               sti
10911 00002A3E 5E                               pop     si
10912                                   no_readdate:
10913 00002A3F 5D                               pop     bp
10914 00002A40 5A                               pop     dx
```

```
00002A41 59                             pop     cx
00002A42 58                             pop     ax
                                cmosck9:    ; 19/12/2023
00002A43 C3                             retn

                                ; ----------------------------------------------------------------------------

                                ; the following code is written by jack gulley in engineering group.
                                ; cp dos (CP/DOS, OS/2) is changing cmos clock rate for its own purposes
                                ; and if the use cold boot the system to use pc dos while running cp dos,
                                ; the cmos clock rate are still slow which slow down disk operations
                                ; of pc dos which uses cmos clock. pc dos is put this code in msinit
                                ; to fix this problem at the request of cp dos.
                                ;
                                ; the program is modified to be run on msinit. equates are defined
                                ; in cmosequ.inc. this program will be called by cmos_clock_read procedure.
                                ;
                                ;  the following code cmosck is used to insure that the cmos has not
                                ;   had its rate controls left in an invalid state on older at's.
                                ;
                                ;   it checks for an at model byte "fc" with a submodel type of
                                ;   00, 01, 02, 03 or 06 and resets the periodic interrupt rate
                                ;   bits in case post has not done it. this initialization routine
                                ;   is only needed once when dos loads. it should be run as soon
                                ;   as possible to prevent slow diskette access.
                                ;
                                ;   this code exposes one to dos clearing cmos setup done by a
                                ;   resident program that hides and re-boots the system.

                                cmosck:                                 ; check and reset rtc rate bits

                                ;model byte and submodel byte were already determined in msinit.

                                    ; 16/06/2018 - Retro DOS v3.0
                                    ; 19/03/2018 (Model: 0FCh, Sub Model: 01h, REF: AMIBIOS Prog. Guide)

                                    ; 19/12/2023 - Retro DOS v5.0

                                    ; 19/12/2023
                                    ; ds = cs
                                    ;push   ax ; not necessary ; 19/12/2023
                                    ;
00002A44 803E[AF05]FC               cmp     byte [model_byte], 0FCh
                                    ;cmp    byte [cs:model_byte], 0FCh
00002A49 75F8                       jnz     short cmosck9  ; Exit if not an AT model
00002A4B 803E[B005]06               cmp     byte [secondary_model_byte], 6  ; 21/04/2024
                                    ;cmp    byte [cs:secondary_model_byte], 6
                                                            ; Is it 06 for the industral AT ?
00002A50 7407                       jz      short cmosck4  ; Go reset CMOS periodic rate if 06
00002A52 803E[B005]04               cmp     byte [secondary_model_byte], 4
                                    ;cmp    byte [cs:secondary_model_byte], 4
                                                            ; Is it 00, 01, 02, or 03 ?
00002A57 73EA                       jnb     short cmosck9  ; EXIT if problem fixed by POST
                                                            ; Also,Secondary_model_byte = 0
                                                            ;   when AH=0C0h, int 15h failed.
                                                            ;       RESET THE CMOS PERIODIC RATE
                                                            ;  Model=FC submodel=00,01,02,03 or 06
                                cmosck4:
00002A59 B08A                       mov     al, 8Ah         ; cmos_reg_a|nmi
                                                            ; NMI disabled on return
00002A5B B426                       mov     ah, 26h         ; 00100110b
                                                            ; Set divider & rate selection
00002A5D E80B00                     call    cmos_write
00002A60 B08B                       mov     al, 8Bh         ; cmos_reg_b|nmi
                                                            ; NMI disabled on return
00002A62 E82000                     call    cmos_read
00002A65 2407                       and     al, 7           ; 00000111b
                                                            ; clear SET,PIE,AIE,UIE,SQWE
00002A67 88C4                       mov     ah, al
00002A69 B00B                       mov     al, 0Bh         ; cmos_reg_b
                                                            ; NMI enabled on return
                                    ; 19/12/2023
                                    ;call   cmos_write
                                ;cmosck9:
                                    ;pop    ax ; 19/12/2023
                                    ;retn

                                    ; 19/12/2023
                                    ;jmp    short cmos_write

                                ; ============== S U B   R O U T I N E =======================================

                                ;--- cmos_write ------------------------------------------------------------
                                ;           write byte to cmos system clock configuration table       :
                                ;                                                                     :
                                ; input: (al)=     cmos table address to be written to                    :
                                ;           bit     7 = 0 for nmi enabled and 1 for nmi disabled on exit    :
                                ;           bits 6-0 = address of table location to write                   :
                                ;     (ah)= new value to be placed in the addressed table location         :
                                ;                                                                     :
                                ; output:   value in (ah) placed in location (al) with nmi left disabled   :
                                ;           if bit 7 of (al) is on. during the cmos update both nmi and    :
                                ;           normal interrupts are disabled to protect cmos data integrity. :
                                ;           the cmos address register is pointed to a default value and    :
                                ;           the interrupt flag restored to the entry state on return.      :
                                ;           only the cmos location and the nmi state is changed.             :
                                ;-----------------------------------------------------------------------------

                                cmos_write:                             ; write (ah) to location (al)
00002A6B 9C                                 pushf                       ;
00002A6C 50                                 push    ax                  ; save work register values
00002A6D FA                                 cli
00002A6E 50                                 push    ax                  ; save user nmi       state
00002A6F 0C80                               or      al, 80h             ; disable nmi for us
00002A71 E670                               out     70h, al             ; CMOS Memory/RTC Index       Register:
                                                                        ; RTC Seconds
00002A73 90                                 nop
00002A74 88E0                               mov     al, ah
00002A76 E671                               out     71h, al             ; CMOS Memory/RTC Data Register
00002A78 58                                 pop     ax                  ; get user nmi
00002A79 2480                               and     al, 80h
00002A7B 0C0F                               or      al, 0Fh
00002A7D E670                               out     70h, al             ; CMOS Memory/RTC Index       Register:
                                                                        ; RTC Seconds
00002A7F 90                                 nop
00002A80 E471                               in      al, 71h             ; CMOS Memory/RTC Data Register
00002A82 58                                 pop     ax                  ; restore work registers

                                            ; 19/12/2023
                                            ;push   cs                  ; *place code segment in stack and
                                            ;call   cmos_popf           ; *handle popf for b- level 80286
                                            ;retn
00002A83 EB18                               jmp     short cmos_rw_popf
```

```
                                ; ============== S U B   R O U T I N E =======================================
                                ;--- CMOS_READ -----------------------------------------------------------------
                                ;           read byte from cmos system clock configuration table             :
                                ;                                                                            :
                                ; input: (al)=     cmos table address to be read                            :
                                ;           bit    7 = 0 for nmi enabled and 1 for nmi disabled on exit    :
                                ;           bits 6-0 = address of table location to read                   :
                                ;                                                                          :
                                ; output: (al)     value at location (al) moved into (al). if bit 7 of (al) was   :
                                ;           on then nmi left disabled. during the cmos read both nmi and   :
                                ;           normal interrupts are disabled to protect cmos data integrity. :
                                ;           the cmos address register is pointed to a default value and    :
                                ;           the interrupt flag restored to the entry state on return.      :
                                ;           only the (al) register and the nmi state is changed.             :
                                ;------------------------------------------------------------------------------

                                cmos_read:                      ; read location (al) into (al)
00002A85 9C                             pushf
00002A86 FA                             cli
00002A87 53                             push   bx
                                        ;push  ax ; *           ; AL = cmos table address to be read
                                        ; 19/12/2023
00002A88 89C3                           mov    bx, ax ; * ; input
00002A8A 0C80                           or     al, 80h
00002A8C E670                           out    70h, al          ; CMOS Memory/RTC Index      Register:
                                                                ; RTC Seconds
00002A8E 90                             nop                     ; (undocumented delay needed)
00002A8F E471                           in     al, 71h          ; CMOS Memory/RTC Data Register

                                        ;mov   bx, ax ; output
                                        ;pop   ax ; * ; input

                                        ; 19/12/2023
                                        ; al = output, bl = input
00002A91 93                             xchg   ax, bx ; *
                                        ; bl = output, al = input

00002A92 2480                           and    al, 80h
00002A94 0C0F                           or     al, 0Fh
00002A96 E670                           out    70h, al          ; CMOS Memory/RTC Index      Register:
                                                                ; RTC Seconds
00002A98 90                             nop
00002A99 E471                           in     al, 71h          ; CMOS Memory/RTC Data Register
                                        ;mov   ax, bx ; * ; output
                                        ; 19/12/2023
00002A9B 93                             xchg   ax, bx
00002A9C 5B                             pop    bx

                                        ; 19/12/2023
                                cmos_rw_popf:
00002A9D 0E                             push   cs               ; *place code segment in stack and
00002A9E E80100                         call   cmos_popf        ; *handle popf for b- level 80286
00002AA1 C3                             retn                    ; return with flags restored

                                ; ---------------------------------------------------------------------------

                                cmos_popf:
00002AA2 CF                             iret                    ; popf for level b- parts
                                                                ; return far and restore flags

                                ; ---------------------------------------------------------------------------
                                ; MSINIT.ASM (MSDOS 6.0, 1991)
                                ; ---------------------------------------------------------------------------
                                ; The following routines provide support for reading in the file MSDOS.SYS.
                                ; ---------------------------------------------------------------------------

                                ; 15/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ;
                                ; (For Retro DOS, 'IO.SYS' and 'MSDOS.SYS' are already loaded together
                                ;  at once -as single kernel file- by the Retro DOS boot sector code.
                                ;  So, following disk reads -MSDOS.SYS loading- is not needed!
                                ;  Only needing is to move MSDOS Kernel to it's final memory location.)

                                ; 20/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
                                ; ---------------------------------------------------------------------------

                                ;ClusterH:  dw 0  ; 20/12/2023

                                ; ============== S U B   R O U T I N E =======================================

                                ; GetClus, read in a cluster at a specified address
                                ;
                                ;  bx = cluster to read
                                ;  cx = sectors per cluster
                                ;  es:di = load location

                                ; 17/10/2022
                                ;DISKRD equ diskrd - DOSBIOSEG_2C7h      ; (8E5h for MSDOS 5.0 IO.SYS)
                                ; 09/12/2022
                                DISKRD equ diskrd

                                        ; 29/12/2023
                                        ; 20/12/2023 - Retro DOS v5.0
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2DC4h)

                                        ; si:bx = (32 bit) cluster to read
                                        ; cx = sectors per cluster
                                        ; es:di = load location

                                ; 06/04/2024
                                %if 1
                                        ; 17/10/2022
                                getclus:
                                        ; 12/12/2023
                                        ; ds = cs

00002AA3 51                             push   cx ; 1*
00002AA4 57                             push   di ; 2*
                                        ;mov   [cs:doscnt], cx
00002AA5 890E[001A]                     mov    [doscnt], cx ; 12/12/2023

                                        ; 20/12/2023
                                        ;;mov  [cs:ClusterH], si ; high word of cluster number
                                        ;mov   [ClusterH], si ; high word of cluster number
00002AA9 89F5                           mov    bp, si

00002AAB 89D8                           mov    ax, bx

                                        ;dec   ax
                                        ;dec   ax
                                        ; 20/12/2023
00002AAD 83E802                         sub    ax, 2
```

```
                                        ;;sbb   [cs:ClusterH], 0
                                        ;sbb    [ClusterH], 0
00002AB0 83DD00                         sbb     bp, 0

                                        ; 20/12/2023
                                        ;;xchg  ax, [cs:ClusterH]
                                        ;xchg   ax, [ClusterH]
00002AB3 95                             xchg    ax, bp

00002AB4 F7E1                           mul     cx

                                        ;;xchg  ax, [cs:ClusterH]
                                        ;xchg   ax, [ClusterH]
00002AB6 95                             xchg    ax, bp ; (+)
                                        ;
00002AB7 F7E1                           mul     cx              ;; convert to logical sector
                                                                ;; dx:ax = matching logical sector number
                                                                ;;       starting from the data sector
                                        ;;add   ax, [cs:bios_l]
                                        ;;adc   dx, [cs:bios_h]       ; dx:ax = first logical sector to read
                                        ; 12/12/2023
                                        ;add    ax, [bios_l]
                                        ;adc    dx, [bios_h]   ; dx:ax = first logical sector to read

                                        ; 20/12/2023
                                        ;;add   dx, [cs:ClusterH]
                                        ;add    ax, [cs:First_Data_Sector]
                                        ;adc    dx, [cs:First_Data_Sector+2]
00002AB9 01EA                           add     dx, bp ; (+)
                                        ;add    dx, [ClusterH] ; convert to logical sector
                                                                ; dx:ax = matching logical sector number
                                                                ;       starting from the data sector
00002ABB 0306[FC19]                     add     ax, [First_Data_Sector]
00002ABF 1316[FE19]                     adc     dx, [First_Data_Sector+2]
                                                                ; dx:ax = first logical sector to read
                                unpack:
                                        ; 20/12/2023
00002AC3 1E                             push    ds ; 3* ; ds = cs ; 12/12/2023
00002AC4 52                             push    dx ; 4* ; * ; 12/12/2023
00002AC5 50                             push    ax ; 5*
                                        ; 29/12/2023
00002AC6 56                             push    si ; 6*
00002AC7 53                             push    bx ; 7*

                                        ;;mov   si, [cs:fatloc]
                                        ;mov    si, [fatloc] ; 12/12/2023
                                        ;mov    ds, si
                                        ; 20/12/2023
                                        ;mov    ax, [fatloc]
                                        ;mov    ds, ax
00002AC8 53                             push    bx ; 8*
00002AC9 FF36[021A]                     push    word [fatloc] ; 9*

                                        ;test   byte [cs:fbigfat], 20h
00002ACD F606[061A]20                   test    byte [fbigfat], 20h   ; fbigbig FAT32 ?
00002AD2 1F                             pop     ds ; 9* ; ds = [fatloc]
00002AD3 7415                           jz      short not_32bit_cluster ; no
                                unpack32:
                                        ;push   dx
00002AD5 89F2                           mov     dx, si
                                        ;mov    si, bx
00002AD7 5E                             pop     si ; 8* ; si = bx
00002AD8 01F6                           add     si, si
00002ADA 11D2                           adc     dx, dx
00002ADC 01F6                           add     si, si
00002ADE 11D2                           adc     dx, dx
                                                                ; dx:si = 4*(si:bx) ; clust num offset from FAT entry 0
00002AE0 E89600                         call    get_fat_sector
00002AE3 8B7702                         mov     si, [bx+2]      ; high word of the FAT32 cluster number
00002AE6 8B1F                           mov     bx, [bx]        ; low word of the FAT32 cluster number
                                        ;pop    dx
00002AE8 EB45                           jmp     short getcl1

                                not_32bit_cluster:
                                        ;mov    si, bx          ; next cluster
00002AEA 5E                             pop     si ; 8* ; si = bx
00002AEB 2EF606[061A]40                 test    byte [cs:fbigfat], 40h; fbig
                                                                ; 16 bit fat?
00002AF1 752F                           jnz     short unpack16 ; yes
                                unpack12:
00002AF3 D1EE                           shr     si, 1           ; 12 bit fat. si = si/2
                                                                ; si = clus + clus/2
00002AF5 01DE                           add     si, bx          ;
                                                                ; (si = byte offset of the cluster in the FAT)
                                        ;push   dx ; 12/12/2023
00002AF7 31D2                           xor     dx, dx
                                        ; 12/12/2023
                                        ; ds = FAT buffer segment
00002AF9 E87D00                         call    get_fat_sector
                                        ;pop    dx ; 12/12/2023

00002AFC 8B07                           mov     ax, [bx]        ; save it into ax
00002AFE 750C                           jnz     short even_odd ; if not a splitted fat, check even-odd.
                                        ; 25/06/2023
                                        ;mov    al, [bx]        ; splitted fat

                                        ; 12/12/2023
                                        ;mov    [cs:temp_cluster], al
00002B00 50                             push    ax ; **; al = low 8 bits of 12 bits cluster number

00002B01 46                             inc     si              ; (next byte)

                                        ;push   dx ; 12/12/2023
00002B02 31D2                           xor     dx, dx
00002B04 E87200                         call    get_fat_sector
                                        ;pop    dx ; 12/12/2023

                                        ;mov    al, ds:0
                                        ; 12/12/2023
                                        ; ds = FAT buffer segment
                                        ;mov    al, [0] ; 19/10/2022
                                        ;mov    [cs:temp_cluster+1], al
                                        ;mov    ax, [cs:temp_cluster]
                                        ; 12/12/2023
                                        ;mov    al, [cs:temp_cluster]
00002B07 58                             pop     ax  ; ** ; al = low 8 bits of 12 bits cluster number
00002B08 8A260000                       mov     ah, [0] ; high 4 bits (bits 7 to 11) of 12 bits cluster num
                                even_odd:
                                        ; 29/12/2023
00002B0C 5B                             pop     bx ; 7*         ; restore old fat entry       value
00002B0D 53                             push    bx              ; save it right       away.
00002B0E D1EB                           shr     bx, 1           ; was it even or odd?
00002B10 7308                           jnc     short havclus   ; it was even.
00002B12 D1E8                           shr     ax, 1           ; odd. massage fat value and keep
```

```
                                                        ; the highest 12 bits.
00002B14 D1E8                           shr     ax, 1
00002B16 D1E8                           shr     ax, 1
00002B18 D1E8                           shr     ax, 1
                                havclus:
00002B1A 89C3                           mov     bx, ax          ; now bx = new fat entry.
00002B1C 81E3FF0F                       and     bx, 0FFFh       ; keep low 12 bits.
00002B20 EB0B                           jmp     short unpackx
                                ; ---------------------------------------------------------------------------

                                unpack16:
                                        ;push   dx      ; 12/12/2023
00002B22 31D2                           xor     dx, dx ; 0
00002B24 D1E6                           shl     si, 1           ; extend to 32 bit offset
                                        ;adc    dx, 0
                                        ; 12/12/2023
00002B26 D1D2                           rcl     dx, 1

                                        ; 12/12/2023
                                        ; ds = FAT buffer segment
00002B28 E84E00                         call    get_fat_sector
                                        ;pop    dx      ; 12/12/2023
00002B2B 8B1F                           mov     bx, [bx]        ;
                                                        ; bx = new fat entry.
                                unpackx:
                                        ; 20/12/2023
00002B2D 31F6                           xor     si, si          ; high word of cluster number = 0
                                                        ; (FAT12 or FAT16)
                                getcl1:
                                        ; 29/12/2023
00002B2F 58                             pop     ax      ; 7* - cluster number lw
                                        ;pop    word [cs:ClusterH]
00002B30 5A                             pop     dx      ; 6* - cluster number hw

                                        ; 20/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                        ; (this is a fast kernel loading method by the MSDOS programmer)
                                        ; ((consequtive clusters --> consequtive sectors))

00002B31 29D8                           sub     ax, bx  ; previous - current (or current - new)
                                        ;sbb    [cs:ClusterH], si
00002B33 19F2                           sbb     dx, si
                                        ;;cmp   [cs:ClusterH], -1 ; one apart? (current = previous+1)
                                        ;cmp    dx, -1
                                        ; 29/12/2023
00002B35 42                             inc     dx ; -1 -> 0
00002B36 7501                           jnz     short not_consequental
                                        ;cmp    ax, -1          ; 0FFFFh ; is [ClusterH]:ax = -1 ?
00002B38 40                             inc     ax ; -1 -> 0
                                not_consequental:
00002B39 58                             pop     ax ; 5*         ; restore logical sector (low)
00002B3A 5A                             pop     dx ; 4* ; * ; 12/12/2023
00002B3B 1F                             pop     ds ; 3*

                                        ;; 12/12/2023
                                        ;; (this is a fast kernel loading method by the MSDOS programmer)
                                        ;; ((consequtive clusters --> consequtive sectors))
                                        ;; ds = cs
                                        ;sub    si, bx
                                        ;cmp    si, -1          ; one apart? (consequtive?)
                                        ;                       ; (current = previous+1)

00002B3C 7507                           jnz     short getcl2    ; no, read [doscnt] sectors

                                        ;add    [cs:doscnt], cx ; (cx = sectors per cluster)
00002B3E 010E[001A]                     add     [doscnt], cx ; 12/12/2023 ; add to read count
00002B42 E97EFF                         jmp     unpack
                                ; ---------------------------------------------------------------------------

                                getcl2:
00002B45 56                             push    si ; 20/12/2023
00002B46 53                             push    bx
                                        ; bx = low word of the new cluster number
                                        ; 20/12/2023 - Retro DOS v5.0 (32 bit cluster numbers)
                                        ; si = high word of the new cluster number
00002B47 52                             push    dx              ; sector to read (high word)
00002B48 50                             push    ax              ; sector to read (low word)

                                        ; 12/12/2023
                                        ; ds = cs
                                        ;mov    ax, [cs:drvfat]         ; get drive and       fat spec
                                        ;mov    cx, [cs:doscnt]
00002B49 A1[FA19]                       mov     ax, [drvfat]    ; get drive and       fat spec

                                        ;;;
                                        ; 20/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                        ;
                                        ; dma and segment (64K boundary) overrun precaution
                                        ; (sector count will be decreased if it is required)

00002B4C 89F9                           mov     cx, di
00002B4E F7D1                           not     cx              ; cx = 65535 - cx
00002B50 D1E9                           shr     cx, 1           ; cx = cx/2
00002B52 30C9                           xor     cl, cl
00002B54 86E9                           xchg    cl, ch          ; cx = cx/256

                                        ;cmp    cx, [cs:doscnt]
                                                                ; if sector read count > cx, decrease it to cx
00002B56 3B0E[001A]                     cmp     cx, [doscnt]
00002B5A 7604                           jbe     short getcl3
                                        ;;;
                                        ;mov    cx, [cs:doscnt]
00002B5C 8B0E[001A]                     mov     cx, [doscnt]
                                getcl3:
00002B60 5A                             pop     dx              ; sector to read for diskrd (low)
                                        ;pop    word [cs:start_sec_h]
                                        ; 12/12/2023
00002B61 8F06[9C04]                     pop     word [start_sec_h]
                                                                ; sector to read for diskrd (high)
                                        ; 06/04/2024
                                        ;;;
00002B65 51                             push    cx ; +*
                                        ;;;

                                        ; 12/12/2023
                                        ; ds = cs
                                        ;push   ds
                                        ;push   cs
                                        ;pop    ds

00002B66 0E                             push    cs              ; simulate far call

                                        ; 20/12/2023
                                        ; 17/10/2022
00002B67 BD[0C0A]                       mov     bp, DISKRD      ; offset diskrd
                                        ;mov    bp, 0A2Bh       ; 20/12/2023
```

```
                                        ;       (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0A2Bh ; 364h:0A2Bh)
                                        ;mov    bp, 8E5h        ; 17/10/2022
                                                                ; 2C7h:8E5h = 70h:2E55h

00002B6A E804EF                         call    call_bios_code ; read the clusters

                                        ;pop    ds
                                        ; 12/12/2023
                                        ; ds = cs

                                        ; 06/04/2024
                                        ;;;
00002B6D 58                             pop     ax ; +*         ; sector count
                                        ;;;

00002B6E 5B                             pop     bx              ; lw of the new cluster number
00002B6F 5E                             pop     si ; 20/12/2023 ; hw of the new cluster number

00002B70 5F                             pop     di ; 2* - (kernel) load location (es:di)

                                        ; 06/04/2024
                                        ;;mov   ax, [cs:doscnt]       ; get number of       sectors read
                                        ;; 12/12/2023
                                        ;mov    ax, [doscnt]

00002B71 86C4                           xchg    ah, al          ; multiply by 256
00002B73 D1E0                           shl     ax, 1           ; times 2 equal 512
00002B75 01C7                           add     di, ax          ; update load location

00002B77 59                             pop     cx ; 1*         ; restore sectors/cluster

00002B78 C3                             retn

                                ; =============== S U B   R O U TI N E =======================================

                                ;function: find and read the corresponding fat sector into ds:0
                                ;
                                ;in). dx:si - offset value (starting from fat entry 0) of fat entry to find. M054
                                ;       ds - fatloc segment
                                ;       cs:drvfat - logical drive number, fat id
                                ;       cs:md_sectorsize
                                ;       cs:last_fat_secnum - last fat sector number read in.
                                ;
                                ;out). corresponding fat sector read in.
                                ;       bx = offset value from fatlog segment.
                                ;       other registers are saved.
                                ;       zero flag set if the fat entry is splitted, i.e., when 12 bit fat entry
                                ;       starts at the last byte of the fat sector. in this case, the caller
                                ;       should save this byte, and read the next fat sector to get the rest
                                ;       of the fat entry value. (this will only happen with the 12 bit fat.)

                                        ; 17/10/2022
                                get_fat_sector:
                                        ; 20/12/2023
                                        ; 12/12/2023
                                        ; ds = fat buffer segment

                                        ; 12/12/2023
                                        ;push   ax ; (not necessary)
00002B79 51                             push    cx ; read count (sectors per cluster)
00002B7A 57                             push    di ; IBMDOS.COM/MSDOS.SYS load offset
00002B7B 56                             push    si ; FAT offset value (from fat entry 0)
00002B7C 06                             push    es ; IBMDOS.COM/MSDOS.SYS load segment
00002B7D 1E                             push    ds ; FAT buffer segment

                                        ; 12/12/2023
00002B7E 0E                             push    cs
00002B7F 1F                             pop     ds

                                        ; 06/04/2024
                                        ; dx:si = offset value (starting from fat entry 0)
                                        ;             of fat entry to find

00002B80 89F0                           mov     ax, si
                                        ;;mov   cx, [cs:md_sectorsize] ; 512
                                        ; 12/12/2023
                                        ;mov    cx, [md_sectorsize] ; 512
                                        ;div    cx              ; ax = sector number, dx = offset
                                        ; 12/12/2023
                                        ;nop

                                        ; 12/12/2023
00002B82 F736[081A]                     div     word [md_sectorsize] ; 512

                                        ; ax = FAT sector (sequence/index) number
                                        ; dx = cluster number offset

                                        ; Get rid of the assumption that
                                        ; there is only one reserved sector

                                        ; 12/12/2023 ; *
                                        ;push   es ; *
                                        ;push   ds ; *
                                        ;push   di ; *
00002B86 50                             push    ax
                                        ;push   cs ; *
                                        ;pop    ds ; *

                                        ;mov    ax, [cs:drvfat]       ; get drive # and FAT id
                                        ; 12/12/2023
00002B87 A1[FA19]                       mov     ax, [drvfat]   ; get drive # and FAT id
00002B8A BD[A405]                       mov     bp, SETDRIVE
                                        ;mov    bp, 5AEh  ; PCDOS 7.1 IBMBIO.COM
                                        ;;mov   bp, 4D7h        ; setdrive
                                                                ; at 2C7h:4D7h = 70h:2A47h
00002B8D 0E                             push    cs              ; simulate far call
00002B8E E8E0EE                         call    call_bios_code ; get bds for drive
00002B91 58                             pop     ax              ; (sector number -without reserved and hidden sectors-)
00002B92 26034509                       add     ax, [es:di+9]  ; [es:di+BDS.resectors]
                                                                ; add #reserved_sectors
                                        ; 12/12/2023
                                        ;pop    di ; *
                                        ;pop    ds ; *
                                        ;pop    es ; *

                                        ; 12/12/2023
                                        ; ds = cs
00002B96 3B06[0A1A]                     cmp     ax, [last_fat_sec_num]
                                        ;cmp    ax, [cs:last_fat_sec_num]
00002B9A 741C                           jz      short gfs_split_chk ; don't need to read it again.
00002B9C A3[0A1A]                       mov     [last_fat_sec_num], ax
                                        ;mov    [cs:last_fat_sec_num], ax
                                                                ; sector number
                                                                ; (in the partition, without hidden sectors)
```

```
11535                                   ; 13/12/2023
11536 00002B9F 07                       pop     es ; FAT buffer segment (DS on top of the stack)
11537 00002BA0 06                       push    es ; (put it on top of the stack again)
11538
11539 00002BA1 52                       push    dx ; cluster number offset
11540
11541                                   ; 12/12/2023
11542 00002BA2 31C9                     xor     cx, cx
11543 00002BA4 890E[9C04]               mov     [start_sec_h], cx ;0
11544                                   ;mov    word [cs:start_sec_h], 0
11545                                                   ; prepare to read the fat sector
11546                                                   ; start_sec_h is always       0 for fat sector.
11547 00002BA8 89C2                     mov     dx, ax
11548                                   ; 12/12/2023
11549 00002BAA 41                       inc     cx ; cx = 1
11550                                   ;mov    cx, 1           ; 1 sector read
11551                                   ;mov    ax, [cs:drvfat]
11552 00002BAB A1[FA19]                 mov     ax, [drvfat]
11553                                   ;push   ds
11554                                   ;pop    es
11555
11556 00002BAE 31FF                     xor     di, di  ; 0     ; es:di -> fatloc segment:0
11557
11558                                   ; 12/12/2023
11559                                   ;push   ds
11560                                   ;push   cs
11561                                   ;pop    ds
11562
11563 00002BB0 0E                       push    cs              ; simulate far call
11564
11565                                   ; 20/12/2023
11566                                   ; 17/10/2022
11567 00002BB1 BD[0C0A]                 mov     bp, DISKRD      ; offset diskrd
11568                                   ;mov    bp, 0A2Bh       ; 20/12/2023
11569                                   ;       (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0A2Bh ; 364h:0A2Bh)
11570                                   ;mov    bp, 8E5h        ; 17/10/2022
11571                                                   ; 2C7h:8E5h = 70h:2E55h
11572
11573 00002BB4 E8BAEE                                   call    call_bios_code ; read the clusters
11574
11575                                                   ; 12/12/2023
11576                                                   ;pop    ds
11577                                                   ; ds = cs = biosdata segment
11578
11579 00002BB7 5A                                       pop     dx ; cluster number offset
11580
11581                                   gfs_split_chk:
11582                                                   ; 13/12/2023
11583                                                   ;mov    cx, [cs:md_sectorsize] ; 512
11584 00002BB8 8B0E[081A]                               mov     cx, [md_sectorsize]
11585                                   ;gfs_split_chk:
11586 00002BBC 49                                       dec     cx              ; 511
11587 00002BBD 39CA                                     cmp     dx, cx          ; if offset points to the
11588                                                                           ; last byte of this sector,
11589                                                                           ; then splitted       entry.
11590 00002BBF 89D3                                     mov     bx, dx          ; set bx to dx
11591
11592                                                   ; 12/12/2023
11593                                                   ; bx = dx = cluster number offset in the FAT buffer
11594 00002BC1 1F                                       pop     ds ; FAT buffer segment
11595 00002BC2 07                                       pop     es ; IBMDOS.COM/MSDOS.SYS load segment
11596 00002BC3 5E                                       pop     si ; FAT offset value (from fat entry 0)
11597 00002BC4 5F                                       pop     di ; IBMDOS.COM/MSDOS.SYS load offset
11598 00002BC5 59                                       pop     cx ; read count (sectors per cluster)
11599                                                   ;pop    ax
11600
11601 00002BC6 C3                                       retn
11602
11603                                   %else
11604
11605                                       ; 06/04/2024 - temporary
11606                                   temp_cluster:       dw      0
11607
11608                                   ClusterH:   dw      0
11609
11610                                   ; =============== S U B R O U T I N E =======================================
11611
11612                                   getclus:
11613                                                   push    cx              ; 1*
11614                                                                           ; si:bx = (32 bit) cluster to read
11615                                                                           ; cx = sectors per cluster
11616                                                                           ; es:di = load location
11617                                                   push    di              ; 2*
11618                                                   mov     [cs:doscnt], cx
11619                                                   mov     ax, bx
11620                                                   mov     [cs:ClusterH], si ; high word of cluster number
11621                                                   sub     ax, 2
11622                                                   sbb     word [cs:ClusterH], 0
11623                                                   xchg    ax, [cs:ClusterH]
11624                                                   mul     cx
11625                                                   xchg    ax, [cs:ClusterH]
11626                                                   mul     cx
11627                                                   add     dx, [cs:ClusterH] ; convert to logical sector
11628                                                                           ; dx:ax = matching logical sector number
11629                                                                           ; starting from the data sector
11630                                                   add     ax, [cs:First_Data_Sector]
11631                                                   adc     dx, [cs:First_Data_Sector+2]
11632                                                                           ; dx:ax = first logical sector to read
11633                                   unpack:
11634                                                   push    ds              ; 3*
11635                                                   push    ax              ; 4*
11636                                                   push    si              ; 5*
11637                                                   push    bx              ; 6*
11638                                                   mov     ax, [cs:fatloc]
11639                                                   mov     ds, ax
11640                                                   test    byte [cs:fbigfat], 20h ; fbigbig
11641                                                                           ; FAT32 ?
11642                                                   jz      short not_32bit_cluster ; no
11643                                   unpack32:                               ; yes
11644                                                   push    dx              ; 7*
11645                                                   mov     dx, si
11646                                                   mov     si, bx
11647                                                   add     si, si
11648                                                   adc     dx, dx
11649                                                   add     si, si
11650                                                   adc     dx, dx          ; dx:si = 4*(si:bx)
11651                                                   call    get_fat_sector
11652                                                   mov     si, [bx+2]      ; byte 16-31 of the FAT32 cluster number
11653                                                   mov     bx, [bx]        ; byte 0-15 of the FAT32 cluster number
11654                                                   pop     dx              ; 7*
11655                                                   jmp     short getcl1
11656                                   ; ---------------------------------------------------------------------------
11657
11658                                   not_32bit_cluster:
```

```
                        mov     si, bx          ; next cluster
                        test    byte [cs:fbigfat], 40h ; fbig
                                                ; FAT16 ?
                        jnz     short unpack16  ; yes
unpack12:
                        shr     si, 1
                        add     si, bx          ; 12 bit fat. si=si/2
                                                ; si = clus + clus/2
                                                ; (si = byte offset of the cluster in the FAT)
                        push    dx
                        xor     dx, dx
                        call    get_fat_sector
                        pop     dx
                        mov     ax, [bx]        ; save cluster number into ax
                        jnz     short even_odd  ; if not a splitted fat, check even-odd
                        mov     al, [bx]        ; (not needed!) Erdogan Tan - 2023
                        mov     byte [cs:temp_cluster], al ; splitted fat
                        inc     si              ; (next byte)
                        push    dx
                        xor     dx, dx
                        call    get_fat_sector
                        pop     dx
                        mov     al, [0]          ; mov ah,[0]
                        mov     byte [cs:temp_cluster+1], al
                        mov     ax, [cs:temp_cluster] ; mov al,[cs:temp_cluster]
even_odd:
                        pop     bx              ; restore old fat entry value
                        push    bx              ; 6*
                        shr     bx, 1           ; was it even or odd?
                        jnb     short havclus   ; it was even
                        shr     ax, 1           ; odd. massage fat value and keep
                                                ; the highest 12 bits.
                        shr     ax, 1
                        shr     ax, 1
                        shr     ax, 1
havclus:
                        mov     bx, ax          ; now bx = new fat entry
                        and     bx, 0FFFh       ; keep low 12 bits
                        jmp     short unpackx
                ; ---------------------------------------------------------------------------

                unpack16:
                        push    dx
                        xor     dx, dx          ; extend to 32 bit offset
                        shl     si, 1           ; cluster number * 2
                        adc     dx, 0
                        call    get_fat_sector
                        pop     dx
                        mov     bx, [bx]        ; bx = new fat entry
                unpackx:
                        xor     si, si          ; high word of cluster number = 0
                                                ; (FAT12 or FAT16)
                getcl1:
                        pop     ax              ; 6* - cluster number lw
                        pop     word [cs:ClusterH] ; 5* - cluster number hw
                        sub     ax, bx          ; previous - current (or current - new)
                        sbb     [cs:ClusterH], si
                        cmp     word [cs:ClusterH], -1 ; one apart? (current = previous+1)
                        jnz     short not_consenquental
                        cmp     ax, -1          ; 0FFFFh ; is [ClusterH]:ax = -1 ?
                not_consenquental:
                        pop     ax              ; 4* - low word of first logical sector
                        pop     ds              ; 3*
                        jnz     short getcl2
                        add     [cs:doscnt], cx ; consequental cluster read, +1 cluster sectors
                                                ; (cx = sectors per cluster)
                        jmp     unpack
                ; ---------------------------------------------------------------------------

                getcl2:
                        push    bx
                        push    si
                        push    dx              ; sector to read (high)
                        push    ax              ; sector to read (low)
                        mov     ax, [cs:drvfat] ; get drive and fat spec
                        mov     cx, di          ; dma and segment (64K boundary) overrun precaution
                                                ; (sector count will be decreased if it is required)
                        not     cx              ; cx = 65535 - cx
                        shr     cx, 1           ; cx = cx/2
                        xor     cl, cl
                        xchg    cl, ch          ; cx = cx/256
                        cmp     cx, [cs:doscnt] ; if sector read count > cx, decrease it to cx
                        jbe     short getcl3
                        mov     cx, [cs:doscnt]
                getcl3:
                        pop     dx              ; sector to read for diskrd (low)
                        pop     word [cs:start_sec_h] ; sector to read for diskrd (high)
                        push    cx
                        push    ds
                        push    cs
                        pop     ds
                        push    cs              ; simulate far call
                        mov     bp, DISKRD ; BIOSCODE:0A2Bh ; 364h:0A2Bh
                        call    call_bios_code
                        pop     ds
                        pop     ax              ; sector count
                        pop     si
                        pop     bx
                        pop     di              ; 2* - load location (es:di)
                        xchg    ah, al
                        shl     ax, 1           ; ax = ax * 512 ; byte count
                        add     di, ax          ; update load location
                        pop     cx              ; 1* - restore sectors/cluster
                        retn

                ; =============== S U B R O U T I N E =======================================

                get_fat_sector:
                        push    ax              ; dx:si = offset value (starting from fat entry 0)
                                                ;         of fat entry to find
                        push    cx
                        push    di
                        push    si
                        push    es
                        push    ds
                        mov     ax, si
                        mov     cx, [cs:md_sectorsize] ; 512
                        div     cx
                        nop                     ; ax = sector number, dx = offset
                        push    es
                        push    ds
                        push    di
                        push    ax
                        push    cs
```

```
                                        pop     ds
                                        mov     ax, [cs:drvfat] ; get drive # and FAT id
                                        mov     bp, SetDrive ; BIOSCODE:05AEh
                                        push    cs              ; simulate far call
                                        call    call_bios_code  ; get bds for drive
                                        pop     ax              ; (sector number -without reserved and hidden sectors-)
                                        add     ax, [es:di+9]   ; [es:di+BDS.resectors]
                                                                ; add #reserved_sectors
                                        pop     di
                                        pop     ds
                                        pop     es
                                        cmp     ax, [cs:last_fat_sec_num]
                                        jz      short gfs_split_chk ; don't need to read it again
                                        mov     [cs:last_fat_sec_num], ax ; sector number
                                                                ; (in the partition, without hidden sectors)
                                        push    dx
                                        mov     word [cs:start_sec_h], 0 ; prepare to read the fat sector
                                                                ; start_sec_h is always 0 for fat sector
                                        mov     dx, ax
                                        mov     cx, 1           ; 1 sector read
                                        mov     ax, [cs:drvfat]
                                        push    ds
                                        pop     es
                                        xor     di, di          ; es:di -> fatloc segment:0
                                        push    ds
                                        push    cs
                                        pop     ds
                                        push    cs              ; simulate far call
                                        mov     bp, DISKRD ; BIOSCODE:0A2Bh ; 364h:0A2Bh
                                        call    call_bios_code
                                        pop     ds
                                        pop     dx
                                        mov     cx, [cs:md_sectorsize] ; 512
                                gfs_split_chk:
                                        dec     cx              ; 511
                                        cmp     dx, cx          ; if offset points to the last byte of this sector,
                                                                ; then splitted entry.
                                        mov     bx, dx          ; offset value from fatloc segment
                                        pop     ds
                                        pop     es
                                        pop     si
                                        pop     di
                                        pop     cx
                                        pop     ax
                                        retn

                                %endif

                                ; 15/10/2022
                                ;Bios_Data_Init    ends

                                ; ---------------------------------------------------------------------------

                                            ; 09/12/2022
                                            ;db 0

                                numbertodiv equ ($-BData_start)
                                numbertomod equ (numbertodiv % 16)

                                %if numbertomod>0 & numbertomod<16
00002BC7 00<rep 9h>                         times (16-numbertomod) db 0
                                %endif

                                ;align 16

                                ; 09/12/2022
                                IOSYSCODESEGOFF    equ $ - BData_start
                                IOSYSCODESEG       equ (IOSYSCODESEGOFF>>4)+(700h>>4)

                                ;--- End of DOSBIOS data segment ----------------------------------------
                                ; ---------------------------------------------------------------------------
                                            ;db 4 dup(0)
                                ; 09/12/2022
                                ;           times 4 db 0   ; 19/10/2022
                                ; ---------------------------------------------------------------------------

                                ;============================================================================
                                ; DOS BIOS (IO.SYS) CODE SEGMENT
                                ;============================================================================
                                ; 09/12/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ; 30/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)

                                section .BIOSCODE vstart=0

                                ; 30/12/2022
                                ; (BIOSCODE SEGMENT is 2CCh for MSDOS 6.21 IO.SYS)  -- ((25C0h+700h)>>4) --

                                BCode_start:        ; 09/12/2022

                                ; 02/10/2022

                                ;--- DOSBIOS code segment -----------------------------------------------
                                ;------------------------------------------------------------------------
                                ; MSBIO1.ASM (MSDOS 6.0, 1991)
                                ;------------------------------------------------------------------------

                                DOSBIOSEG_2C7h:    ;db 30h dup(0)        ; SEGMENT 2C7h (2C70h-700h=2570h)
00000000 00<rep 30h>                        times 48 db 0          ; 19/10/2022
00000030 7000                   BiosDataWord:      dw 70h

                                ; 15/10/2022
                                ;BIOSDATAWORD      equ BiosDataWord - DOSBIOSEG_2C7h
                                ; 09/12/2022
                                BIOSDATAWORD       equ BiosDataWord

                                ; ---------------------------------------------------------------------------

                                ; 15/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ; 20/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)

                                ;****************************************************************************
                                ;*                                                                          *
                                ;*  seg_reinit is called with ax = our new code segment value,              *
                                ;*    trashes di, cx, es                                                    *
                                ;*                                                                          *
                                ;*  cas -- should be made disposable!                                       *
                                ;*                                                                          *
                                ;****************************************************************************

                                    ; 20/09/2023
                                    ; 10/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                    ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0032h

                                _seg_reinit:
```

```
11907 00000032 2E8E06[3000]                     mov     es, [cs:BIOSDATAWORD]
11908                                                   ; at 2C7h:30h or 70h:25A0h
11909                                           ;mov    di, (offset cdev+2)
11910 00000037 BF[0406]                         mov     di, cdev+2      ; 19/10/2022
11911                                           ;mov    cx, 4           ; (end_BC_entries - cdev)/4
11912                                           ; 10/08/2023
11913 0000003A B90300                           mov     cx, 3 ; (PCDOS 7.1)
11914                                   _seg_reinit_1:
11915 0000003D AB                               stosw                   ; modify Bios_Code entry points
11916 0000003E 47                               inc     di
11917 0000003F 47                               inc     di
11918 00000040 E2FB                             loop    _seg_reinit_1
11919                                           ; 10/08/2023 (PCDOS 7.1)
11920                                           ; (direct jump to i2f_handler from BIOSDATA:bios_i2f)
11921                                           ; (instead of 'bcode_i2f: dw i2f_handler, IOSYSCODESEG')
11922 00000042 26A3[0800]                       mov     [es:bios_i2f_seg], ax ; actual BIOSCODE segment
11923
11924 00000046 CB                               retf
11925
11926                                   ; ---------------------------------------------------------------------------
11927
11928                                   ; 15/10/2022
11929
11930                                   ;****************************************************************************
11931                                   ;*                                                                          *
11932                                   ;*  chardev_entry - main device driver dispatch routine                     *
11933                                   ;*     called with a dummy parameter block on the stack                     *
11934                                   ;*     dw dispatch_table, dw prn/aux numbers (optional)                     *
11935                                   ;*                                                                          *
11936                                   ;*  will eventually take care of doing the transitions in                 *
11937                                   ;*     out of Bios_Code                                                     *
11938                                   ;*                                                                          *
11939                                   ;****************************************************************************
11940
11941                                           ; 20/09/2023
11942                                   chardev_entry:                          ; 0070h:25B3h =       02C7h:0043h
11943 00000047 56                               push    si
11944 00000048 50                               push    ax
11945 00000049 51                               push    cx
11946 0000004A 52                               push    dx
11947 0000004B 57                               push    di
11948 0000004C 55                               push    bp
11949 0000004D 1E                               push    ds
11950 0000004E 06                               push    es
11951 0000004F 53                               push    bx
11952 00000050 89E5                             mov     bp, sp
11953 00000052 8B7612                           mov     si, [bp+18]     ; get return address (dispatch table)
11954                                           ;;mov   ds, word [cs:0030h]
11955                                           ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:0030h]
11956 00000055 2E8E1E[3000]                     mov     ds, [cs:BIOSDATAWORD] ; 17/10/2022
11957                                           ; 20/09/2023 (PCDOS 7.1 - IBMBIO.COM - BIOSCODE:005Ah)
11958 0000005A C434                             les     si, [si]
11959                                           ;mov    ax, [si+2]      ; get the device number       if present
11960 0000005C 8CC0                             mov     ax, es
11961 0000005E A2[2100]                         mov     [auxnum], al
11962 00000061 8826[8004]                       mov     [printdev], ah
11963                                           ;mov    si, [si]        ; point to the device dispatch table
11964 00000065 C41E[1200]                       les     bx, [ptrsav]    ; get pointer to i/o packet
11965 00000069 268A4701                         mov     al, [es:bx+1]  ; [es:bx+unit] ; al = unit code
11966 0000006D 268A670D                         mov     ah, [es:bx+13] ; [es:bx+media]       ; ah = media descrip
11967 00000071 268B4F12                         mov     cx, [es:bx+18] ; [es:bx+count]       ; cx = count
11968 00000075 268B5714                         mov     dx, [es:bx+20] ; [es:bx+start]       ; dx = start sector
11969                                           ; 17/10/2022
11970 00000079 81FE[6F05]                       cmp     si, DSKTBL
11971                                           ;;cmp   si, 579h        ; (PCDOS 7.1 IBMBIO.COM)
11972                                           ;cmp    si, 4A2h        ; dsktbl
11973                                                                   ; at 2C7h:4A2h = 70h:2A12h
11974 0000007D 7517                             jnz     short no_sector32_mapping
11975
11976                                   ; Special case for 32-bit start sector number:
11977                                   ;   if (si==dsktbl) /* if this is a disk device call */
11978                                   ;      set high 16 bits of secnum to 0
11979                                   ;      if (secnum == 0xffff) fetch 32 bit sector number
11980                                   ;
11981                                   ; pass high word of sector number in start_sec_h, low word in dx
11982                                   ;
11983                                   ; note: start_l and start_h are the offsets within the io_request packet
11984                                   ;     which contain the low and hi words of the 32 bit start sector if
11985                                   ;     it has been used.
11986                                   ;
11987                                   ; note: remember not to destroy the registers which have been set up before
11988
11989                                           ; 20/09/2023
11990                                           ;mov    ds:start_sec_h,       0 ; initialize to 0
11991 0000007F C706[9C04]0000                   mov     word [start_sec_h], 0
11992 00000085 83FAFF                           cmp     dx, 0FFFFh
11993 00000088 750C                             jnz     short no_sector32_mapping
11994 0000008A 268B571C                         mov     dx, [es:bx+28] ; [es:bx+start_h]
11995                                                                   ; 32 bits dsk req
11996                                           ;mov    ds:start_sec_h,       dx ; start_sec_h = packet.start_h
11997 0000008E 8916[9C04]                       mov     [start_sec_h], dx
11998 00000092 268B571A                         mov     dx, [es:bx+26] ; [es:bx+start_l]
11999                                                                   ; dx = packet.start_l
12000                                   no_sector32_mapping:
12001 00000096 97                               xchg    ax, di
12002 00000097 268A4702                         mov     al, [es:bx+2]  ; [es:bx+cmd]
12003 0000009B 2E3A04                           cmp     al, [cs:si]
12004 0000009E 732B                             jnb     short command_error
12005 000000A0 98                               cbw                     ; note that al <= 15 means ok
12006 000000A1 D1E0                             shl     ax, 1
12007 000000A3 01C6                             add     si, ax
12008 000000A5 97                               xchg    ax, di
12009 000000A6 26C47F0E                         les     di, [es:bx+14] ; [es:bx+trans]
12010 000000AA FC                               cld
12011                                           ; 17/10/2022
12012 000000AB 2EFF5401                         call    near [cs:si+1]
12013                                           ;call   word ptr cs:si+1
12014 000000AF 7202                             jb      short already_got_ah_status
12015 000000B1 B401                             mov     ah, 1
12016                                   already_got_ah_status:
12017                                           ;;mov   ds, [cs:0030h] ; 15/10/2022
12018                                           ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:0030h]
12019                                                                   ; cas note: shouldn't be needed!
12020 000000B3 2E8E1E[3000]                     mov     ds, [cs:BIOSDATAWORD] ; 17/10/2022
12021                                           ;lds    bx, ds:ptrsav
12022 000000B8 C51E[1200]                       lds     bx, [ptrsav]
12023 000000BC 894703                           mov     [bx+3], ax      ; [bx+status]
12024                                                                   ; mark operation complete
12025 000000BF 5B                               pop     bx
12026 000000C0 07                               pop     es
12027 000000C1 1F                               pop     ds
12028 000000C2 5D                               pop     bp
12029 000000C3 5F                               pop     di
12030 000000C4 5A                               pop     dx
```

```
12031 000000C5 59                               pop     cx
12032 000000C6 58                               pop     ax
12033 000000C7 5E                               pop     si
12034                                           ;add    sp, 2           ; get rid of fake return address
12035                                           ; 20/09/2023 (PCDOS 7.1       - IBMBIO.COM - BIOSCODE:00C8h)
12036 000000C8 44                               inc     sp
12037 000000C9 44                               inc     sp
12038
12039                                           ; fall through into bc_retf
12040                                   ; ---------------------------------------------------------------------------
12041                                   bc_retf:
12042 000000CA CB                               retf
12043                                   ; ---------------------------------------------------------------------------
12044
12045                                   command_error:
12046 000000CB E80700                           call    bc_cmderr
12047 000000CE EBE3                             jmp     short already_got_ah_status
12048                                   ; 15/10/2022
12049                                   ; 01/05/2019
12050
12051                                   ;---------------------------------------------------------------------------
12052                                   ; The following piece of hack is for supporting CP/M compatibility
12053                                   ; Basically at offset 5 we have a far call into 0:c0. But this does not call
12054                                   ; 0:c0 directly instead it call f01d:fef0, because it needs to support 'lhld 6'
12055                                   ; The following hack has to reside at ffff:d0 (= f01d:fef0) if BIOS is loaded
12056                                   ; high.
12057                                   ;---------------------------------------------------------------------------
12058
12059                                           ;db 7 dup(0)
12060
12061                                           ; 20/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
12062                                           ; (PCDOS 7.1 - IBMBIO.COM - BIOSCODE:0D0h)
12063                                           ; 15/10/2022
12064                                           ;dw 0           ; pad to bring offset to 0D0h
12065
12066 000000D0 00<rep 5h>               off_d0:     times 5 db 0   ; 5 bytes from 0:c0 will be copied onto here
12067                                                               ;  which is the CP/M call 5 entry point
12068
12069
12070                                   ; ---------------------------------------------------------------------------
12071
12072                                   ;   exit - all routines return through this path
12073
12074                                   bc_cmderr:
12075 000000D5 B003                             mov     al, 3           ; 2C7h:D5h = 70h:2645h
12076                                                                     ; unknown command error
12077
12078                                   ; =============== S U B   R O U T I N E =======================================
12079
12080                                   ;   now zero the count field by subtracting its current value,
12081                                   ;     which is still in cx, from itself.
12082
12083                                   ;   subtract the number of i/o's NOT YET COMPLETED from total
12084                                   ;     in order to return the number actually complete
12085
12086                                   bc_err_cnt:
12087                                           ;les    bx, ds:ptrsav
12088                                           ; 19/10/2022
12089 000000D7 C41E[1200]                       les     bx, [ptrsav]
12090 000000DB 26294F12                         sub     [es:bx+18], cx ; [es:bx+count]
12091                                                                   ; # of successful i/o's
12092 000000DF B481                             mov     ah, 81h        ; mark error return
12093 000000E1 F9                               stc                    ; indicate abnormal end
12094 000000E2 C3                               retn
12095
12096                                   ; 15/10/2022
12097
12098                                   ;Bios_Code ends
12099
12100                                   ;---------------------------------------------------------------------------
12101                                   ; MSCHAR.ASM - MSDOS 6.0 - 1991
12102                                   ;---------------------------------------------------------------------------
12103                                   ; 15/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
12104                                   ; 10/01/2019 - Retro DOS v4.0
12105
12106                                   ; 30/04/2019
12107
12108                                   ;title     mschar - character and clock devices
12109
12110                                   ;MODE_CTRLBRK      equ     0FFh
12111
12112                                   ; BIOSCODE:00E4h (MSDOS 6.21, IO.SYS)
12113
12114                                   ;****************************************************************************
12115                                   ;*                                                                          *
12116                                   ;*  device driver dispatch tables                                           *
12117                                   ;*                                                                          *
12118                                   ;*  each table starts with a byte which lists the number of                 *
12119                                   ;*  legal functions, followed by that number of words. Each                 *
12120                                   ;*  word represents an offset of a routine in Bios_Code which  *
12121                                   ;*  handles the function. The functions are terminated with                 *
12122                                   ;*  a near return. If carry is reset, a 'done' code is returned             *
12123                                   ;*  to the caller. If carry is set, the ah/al registers are                 *
12124                                   ;*  returned as abnormal completion status. Notice that ds                  *
12125                                   ;*  is assumed to point to the Bios_Data segment throughout.   *
12126                                   ;*                                                                          *
12127                                   ;****************************************************************************
12128
12129                                           ; 20/09/2023
12130                                           ; (PCDOS 7.1 - IBMBIO.COM - BIOSCODE:00E3h)
12131                                           ; 13/12/2022
12132 000000E3 00                               db 0
12133
12134                                           ; 13/12/2022
12135 000000E4 0B                       con_table:  db ((con_table_end - con_table)-1)/2 ; 11
12136                                                                   ; 2C7h:0E4h = 70h:2654h
12137 000000E5 [FA01]                           dw bc_exvec  ; 1FBh     ; bc_exvec at 2C7h:1FBh       = 70h:276Bh
12138                                                                   ; 00 init
12139 000000E7 [FA01]                           dw bc_exvec  ; 1FBh     ; 01
12140 000000E9 [FA01]                           dw bc_exvec  ; 1FBh     ; 02
12141 000000EB [D500]                           dw bc_cmderr ; 0D5h     ; bc_exvec at 2C7h:D5h = 70h:2645h
12142                                                                   ; 03
12143 000000ED [5C01]                           dw con_read  ; 15Ch     ; con_read at 2C7h:15Ch       = 70h:26CCh
12144                                                                   ; 04
12145 000000EF [9F01]                           dw con_rdnd  ; 19Fh     ; con_rdnd at 2C7h:19Fh       = 70h:270Fh
12146                                                                   ; 05
12147 000000F1 [FA01]                           dw bc_exvec  ; 1FBh     ; 06
12148 000000F3 [0802]                           dw con_flush ; 209h     ; con_flush at 2C7h:209h = 70h:2779h
12149                                                                   ; 07
12150 000000F5 [FC01]                           dw con_writ  ; 1FDh     ; con_writ at 2C7h:1FDh       = 70h:276Dh
12151                                                                   ; 08
12152 000000F7 [FC01]                           dw con_writ  ; 1FDh     ; 09
12153 000000F9 [FA01]                           dw bc_exvec  ; 1FBh     ; 0A
12154                                   con_table_end:
```

```
12155 000000FB 1A                       prn_table:  db ((prn_table_end - prn_table)-1)/2 ; 26
12156                                                                       ; 2C7h:0FBh = 70h:266Bh
12157 000000FC [FA01]                               dw bc_exvec   ; 1FBh  ; bc_exvec
12158 000000FE [FA01]                               dw bc_exvec   ; 1FBh  ; 01
12159 00000100 [FA01]                               dw bc_exvec   ; 1FBh  ; 02
12160 00000102 [D500]                               dw bc_cmderr  ;       0D5h    ; bc_cmderr
12161 00000104 [1902]                               dw prn_input  ;       21Ah    ; prn_input
12162                                                                       ; 04 indicate zero chars read
12163 00000106 [C701]                               dw z_bus_exit ; 1C8h  ; z_bus_exit
12164                                                                       ; 05 read non-destructive
12165 00000108 [FA01]                               dw bc_exvec   ; 1FBh  ; 06
12166 0000010A [FA01]                               dw bc_exvec   ; 1FBh  ; 07
12167 0000010C [1E02]                               dw prn_writ   ;       21Fh    ; prn_writ
12168 0000010E [1E02]                               dw prn_writ   ; 21Fh  ; 09
12169 00000110 [4F02]                               dw prn_stat   ; 251h  ; prn_stat
12170 00000112 [FA01]                               dw bc_exvec   ; 1FBh  ; 0B
12171 00000114 [FA01]                               dw bc_exvec   ; 1FBh  ; 0C
12172 00000116 [FA01]                               dw bc_exvec   ; 1FBh  ; 0D
12173 00000118 [FA01]                               dw bc_exvec   ; 1FBh  ; 0E
12174 0000011A [FA01]                               dw bc_exvec   ; 1FBh  ; 0F
12175 0000011C [9402]                               dw prn_tilbusy ; 28Bh ; prn_tilbusy
12176 0000011E [FA01]                               dw bc_exvec   ; 1FBh  ; 11
12177 00000120 [FA01]                               dw bc_exvec   ; 1FBh  ; 12
12178 00000122 [C202]                               dw prn_genioctl ; 2BAh; prn_genioctl
12179 00000124 [FA01]                               dw bc_exvec   ; 1FBh  ; 14
12180 00000126 [FA01]                               dw bc_exvec   ; 1FBh  ; 15
12181 00000128 [FA01]                               dw bc_exvec   ; 1FBh  ; 16
12182 0000012A [FA01]                               dw bc_exvec   ; 1FBh  ; 17
12183 0000012C [FA01]                               dw bc_exvec   ; 1FBh  ; 18
12184 0000012E [F702]                               dw prn_ioctl_query ; 2F0h ; prn_ioctl_query
12185                                   prn_table_end:
12186 00000130 0B                       aux_table:  db ((aux_table_end - aux_table)-1)/2 ; 11
12187                                                                       ; 2C7h:130h = 70h:26A0h
12188 00000131 [FA01]                               dw bc_exvec   ; 1FBh  ; 00 - init
12189 00000133 [FA01]                               dw bc_exvec   ; 1FBh  ; 01
12190 00000135 [FA01]                               dw bc_exvec   ; 1FBh  ; 02
12191 00000137 [D500]                               dw bc_cmderr  ;       0D5h    ; 03
12192 00000139 [1203]                               dw aux_read   ; 30Dh  ; aux_read ; 04       - read
12193 0000013B [3703]                               dw aux_rdnd   ; 335h  ; aux_rdnd - 05       - read non-destructive
12194 0000013D [FA01]                               dw bc_exvec   ; 1FBh  ; 06
12195 0000013F [7803]                               dw aux_flsh   ;       36Ch    ; aux_flsh
12196 00000141 [7F03]                               dw aux_writ   ;       374h    ; aux_writ
12197 00000143 [7F03]                               dw aux_writ   ;       374h    ; 09
12198 00000145 [5703]                               dw aux_wrst   ;       355h    ; aux_wrst
12199                                   aux_table_end:
12200 00000147 0A                       tim_table   db ((tim_table_end - tim_table)-1)/2 ; 10
12201                                                                       ; 2C7h:147h = 70h:26B7h
12202 00000148 [FA01]                               dw bc_exvec   ; 1FBh  ; 00
12203 0000014A [FA01]                               dw bc_exvec   ; 1FBh  ; 01
12204 0000014C [FA01]                               dw bc_exvec   ; 1FBh  ; 02
12205 0000014E [D500]                               dw bc_cmderr  ;       0D5h    ; 03
12206 00000150 [E404]                               dw tim_read   ;       435h    ; tim_read
12207 00000152 [C701]                               dw z_bus_exit ; 1C8h  ; z_bus_exit
12208 00000154 [FA01]                               dw bc_exvec   ; 1FBh  ; 06
12209 00000156 [FA01]                               dw bc_exvec   ; 1FBh  ; 07
12210 00000158 [E503]                               dw tim_writ   ; 3DBh  ; tim_writ
12211 0000015A [E503]                               dw tim_writ   ; 3DBh  ; 09
12212                                   tim_table_end:
12213
12214                                   ; ---------------------------------------------------------------------------
12215
12216                                   ;*************************************************************************
12217                                   ;*                                                                       *
12218                                   ;*  con_read - read cx bytes from keyboard into buffer at es:di          *
12219                                   ;*                                                                       *
12220                                   ;*************************************************************************
12221
12222                                   con_read:                           ; 2C7h:15Ch = 70h:26CCh
12223                                             ;jcxz   short con_exit ; read cx bytes       from keyboard into buffer
12224 0000015C E306                               jcxz    con_exit       ; 19/10/2022
12225                                   con_loop:
12226 0000015E E80500                             call    chrin          ; get char in al
12227 00000161 AA                                 stosb                  ; store char at es:di
12228 00000162 E2FA                               loop    con_loop
12229                                   con_exit:
12230 00000164 F8                                 clc
12231 00000165 C3                                 retn
12232
12233                                   ; =============== S U B   R O U T I N E =======================================
12234
12235                                   ;*************************************************************************
12236                                   ;*                                                                       *
12237                                   ;*  chrin - input single char from keyboard into al                      *
12238                                   ;*                                                                       *
12239                                   ;*    we are going to issue extended keyboard function, if               *
12240                                   ;*    supported. the returning value of the extended keystroke *
12241                                   ;*    of the extended keyboard function uses 0E0h in al                  *
12242                                   ;*    instead of 00h as in the conventional keyboard function. *
12243                                   ;*    this creates a conflict when the user entered real                 *
12244                                   ;*    greek alpha charater (= 0E0h) to  distinguish the extended         *
12245                                   ;*    keystroke and the greek alpha. this case will be handled *
12246                                   ;*    in the following manner:                                           *
12247                                   ;*                                                                       *
12248                                   ;*        ah = 16h                                                       *
12249                                   ;*        int 16h                                                        *
12250                                   ;*        if al == 0, then extended code (in ah)                         *
12251                                   ;*        else if al == 0E0h, then                                       *
12252                                   ;*        if ah <> 0, then extended code (in ah)                         *
12253                                   ;*         else greek_alpha character.                                   *
12254                                   ;*                                                                       *
12255                                   ;*  also, for compatibility reason, if an extended code is             *
12256                                   ;*    detected, then we are going to change the value in al              *
12257                                   ;*    from 0E0h to 00h.                                                  *
12258                                   ;*                                                                       *
12259                                   ;*************************************************************************
12260
12261                                             ; 19/10/2022
12262                                   chrin:
12263 00000166 8A26[7E04]                         mov     ah, [keyrd_func] ; set by msinit. 0 or 10h
12264 0000016A 30C0                               xor     al, al
12265 0000016C 8606[0C00]                         xchg    al, [altah]    ; get character       & zero altah
12266 00000170 08C0                               or      al, al
12267 00000172 752A                               jnz     short keyret
12268 00000174 CD16                               int     16h            ; KEYBOARD -
12269 00000176 09C0                               or      ax, ax
12270 00000178 74EC                               jz      short chrin
12271 0000017A 3D0072                             cmp     ax, 7200h      ; check for ctrl-prtsc
12272 0000017D 7504                               jnz     short alt_ext_chk
12273 0000017F B010                               mov     al, 10h
12274 00000181 EB1B                               jmp     short keyret
12275                                   ; ---------------------------------------------------------------------------
12276
12277                                   ;  if operation was extended function (i.e. keyrd_func != 0) then
12278                                   ;    if character read was 0E0h then
```

```
12279                                   ;      if extended byte was zero (i.e. ah == 0) then
12280                                   ;    goto keyret
12281                                   ;      else
12282                                   ;    set al to zero
12283                                   ;    goto alt_save
12284                                   ;      endif
12285                                   ;    endif
12286                                   ;  endif
12287                                   
12288                                   alt_ext_chk:
12289 00000183 803E[7E04]00                       cmp     byte [keyrd_func], 0
12290 00000188 740C                               jz      short not_ext
12291 0000018A 3CE0                               cmp     al, 0E0h
12292 0000018C 7508                               jnz     short not_ext
12293 0000018E 08E4                               or      ah, ah
12294 00000190 740C                               jz      short keyret
12295 00000192 30C0                               xor     al, al
12296 00000194 EB04                               jmp     short alt_save
12297                                   ; ---------------------------------------------------------------------------
12298                                   
12299                                   not_ext:
12300 00000196 08C0                               or      al, al          ; special case?
12301 00000198 7504                               jnz     short keyret
12302                                   alt_save:
12303 0000019A 8826[0C00]                         mov     [altah], ah     ; store special key
12304                                   keyret:
12305 0000019E C3                                 retn
12306                                   
12307                                   ; ---------------------------------------------------------------------------
12308                                   
12309                                   ;****************************************************************************
12310                                   ;*                                                                          *
12311                                   ;*  con_rdnd - keyboard non destructive read, no wait                       *
12312                                   ;*                                                                          *
12313                                   ;*  pc-convertible-type machine: if bit 10 is set by the dos                *
12314                                   ;*  in the status word of the request packet, and there is no               *
12315                                   ;*  character in the input buffer, the driver issues a system               *
12316                                   ;*  wait request to the rom. on return from the rom, it returns             *
12317                                   ;*  a 'char-not-found' to the dos.                                          *
12318                                   ;*                                                                          *
12319                                   ;****************************************************************************
12320                                   
12321                                             ; 19/10/2022
12322                                   con_rdnd:
12323 0000019F A0[0C00]                           mov     al, [altah]
12324 000001A2 08C0                               or      al, al
12325 000001A4 754C                               jnz     short rdexit
12326 000001A6 8A26[7F04]                         mov     ah, [keysts_func]
12327 000001AA CD16                               int     16h             ; KEYBOARD -
12328 000001AC 751D                               jnz     short gotchr
12329 000001AE 803E[7900]00                       cmp     byte [fhavek09], 0
12330 000001B3 7412                               jz      short z_bus_exit
12331 000001B5 C41E[1200]                         les     bx, [ptrsav]
12332                                             ; 12/12/2022
12333 000001B9 26F6470404                         test    byte [es:bx+4], 04h
12334                                             ;test   word [es:bx+3], 400h ; [es:bx+status]
12335 000001BE 7407                               jz      short z_bus_exit
12336 000001C0 B80041                             mov     ax, 4100h
12337 000001C3 30DB                               xor     bl, bl
12338 000001C5 CD15                               int     15h             ; SYSTEM - WAIT       ON EXTERNAL EVENT (CONVERTIBLE)
12339                                                                     ; AL = condition type, BH = condition compare or mask value
12340                                                                     ; BL = timeout value times 55 milliseconds, 00h     means no timeout
12341                                                                     ; DX = I/O port       address if AL bit 4 set
12342                                   z_bus_exit:
12343 000001C7 F9                                 stc                     ; 2C7h:1C8h = 70h:2738h
12344 000001C8 B403                               mov     ah, 3           ; indicate busy       status
12345 000001CA C3                                 retn
12346                                   ; ---------------------------------------------------------------------------
12347                                   
12348                                   gotchr:
12349 000001CB 09C0                               or      ax, ax
12350 000001CD 7508                               jnz     short notbrk    ; check for null after break
12351 000001CF 8A26[7E04]                         mov     ah, [keyrd_func] ; issue keyboard read function
12352 000001D3 CD16                               int     16h             ; KEYBOARD -
12353 000001D5 EBC8                               jmp     short con_rdnd ; get a real status
12354                                   ; ---------------------------------------------------------------------------
12355                                   
12356                                   notbrk:
12357 000001D7 3D0072                             cmp     ax, 7200h       ; check for ctrl-prtsc
12358 000001DA 7504                               jnz     short rd_ext_chk
12359 000001DC B010                               mov     al, 10h         ; ('P' & 1Fh) ; return control p
12360 000001DE EB12                               jmp     short rdexit
12361                                   ; ---------------------------------------------------------------------------
12362                                   
12363                                   rd_ext_chk:
12364 000001E0 803E[7E04]00                       cmp     byte [keyrd_func], 0 ; extended keyboard function?
12365 000001E5 740B                               jz      short rdexit
12366 000001E7 3CE0                               cmp     al, 0E0h        ; extended key value or       greek alpha?
12367 000001E9 7507                               jnz     short rdexit
12368 000001EB 80FC00                             cmp     ah, 0           ; scan code exist?
12369 000001EE 7402                               jz      short rdexit    ; yes. greek alpha char.
12370 000001F0 B000                               mov     al, 0           ; no. extended key stroke.
12371                                                                     ; change it for       compatibility
12372                                   rdexit:
12373 000001F2 C41E[1200]                         les     bx, [ptrsav]
12374 000001F6 2688470D                           mov     [es:bx+13], al ; [es:bx+media]
12375                                                                     ; return keyboard character here
12376                                   bc_exvec:
12377 000001FA F8                                 clc                     ; bc_exvec at 2C7h:1FBh       = 70h:276Bh
12378                                                                     ; indicate normal termination
12379 000001FB C3                                 retn
12380                                   ; ---------------------------------------------------------------------------
12381                                   
12382                                   ;****************************************************************************
12383                                   ;*                                                                          *
12384                                   ;*  con_write - console write routine                                       *
12385                                   ;*                                                                          *
12386                                   ;*  entry:  es:di -> buffer                                                     *
12387                                   ;*          cx    =  count                                                  *
12388                                   ;*                                                                          *
12389                                   ;****************************************************************************
12390                                   
12391                                   con_writ:
12392                                             ;jcxz   short bc_exvec
12393 000001FC E3FC                               jcxz    bc_exvec        ; 19/10/2022
12394                                             ; 12/12/2022
12395                                             ;jcxz   cc_ret
12396                                   con_lp:
12397 000001FE 268A05                             mov     al, [es:di]
12398 00000201 47                                 inc     di
12399 00000202 CD29                               int     29h             ; DOS 2+ internal - FAST PUTCHAR
12400                                                                     ; AL = character to display
12401 00000204 E2F8                               loop    con_lp
12402                                   cc_ret:
```

```
12403 00000206 F8                              clc
12404 00000207 C3                              retn
12405
12406                                  ; =============== S U B   R O U T I N E =======================================
12407
12408                                  ;****************************************************************************
12409                                  ;*                                                                          *
12410                                  ;*  con_flush - flush out keyboard queue                                    *
12411                                  ;*                                                                          *
12412                                  ;****************************************************************************
12413
12414                                  con_flush:
12415 00000208 C606[0C00]00                    mov     byte [altah], 0       ; clear out holding buffer
12416                                  flloop:                                ; while (charavail()) charread();
12417 0000020D B401                            mov     ah, 1
12418 0000020F CD16                            int     16h             ; KEYBOARD - CHECK BUFFER, DO NOT CLEAR
12419                                                                  ; Return: ZF clear if character      in buffer
12420                                                                  ; AH = scan code, AL = character
12421                                                                  ; ZF set if no character in buffer
12422 00000211 74F3                            jz      short cc_ret
12423 00000213 30E4                            xor     ah, ah
12424 00000215 CD16                            int     16h             ; KEYBOARD - READ CHAR FROM BUFFER, WAIT IF EMPTY
12425                                                                  ; Return: AH = scan code, AL = character
12426 00000217 EBF4                            jmp     short flloop
12427
12428                                  ; ---------------------------------------------------------------------------
12429
12430                                  ; 15/10/2022
12431
12432                                  ;****************************************************************************
12433                                  ;*                                                                          *
12434                                  ;*  some equates for rom bios printer i/o                                       *
12435                                  ;*                                                                          *
12436                                  ;****************************************************************************
12437
12438                                  ; ibm rom status bits (i don't trust them, neither should you)
12439                                  ; warning!!! the ibm rom does not return just one bit. it returns a
12440                                  ; whole slew of bits, only one of which is correct.
12441
12442                                  ;notbusystatus     equ 10000000b         ; not busy
12443                                  ;nopaperstatus     equ 00100000b         ; no more paper
12444                                  ;prnselected       equ 00010000b         ; printer selected
12445                                  ;ioerrstatus       equ 00001000b         ; some kinda error
12446                                  ;timeoutstatus     equ 00000001b         ; time out.
12447                                  ;
12448                                  ;noprinter  equ 00110000b         ; no printer attached
12449
12450                                  ; 18/03/2019 - Retro DOS v4.0
12451                                  ;error_I24_out_of_paper   equ 9 ; MSDOS 6.0, ERR.INC, 1991
12452
12453                                  ; ---------------------------------------------------------------------------
12454
12455                                  ;****************************************************************************
12456                                  ;*                                                                          *
12457                                  ;*  prn_input - return with no error but zero chars read                        *
12458                                  ;*                                                                          *
12459                                  ;*  enter with cx = number of characters requested                          *
12460                                  ;*                                                                          *
12461                                  ;****************************************************************************
12462
12463                                  prn_input:                             ; 2C7h:21Ah = 70h:278Ah
12464 00000219 E8BBFE                          call    bc_err_cnt      ; reset count to zero
12465                                                                  ; (sub reqpkt.count,cx)
12466                                          ; 12/12/2022
12467                                  prn_done:
12468 0000021C F8                              clc                     ; but return with carry      reset for no error
12469 0000021D C3                              retn
12470                                  ; ---------------------------------------------------------------------------
12471
12472                                  ;****************************************************************************
12473                                  ;*                                                                          *
12474                                  ;*  prn_writ - write cx bytes from es:di to printer device                      *
12475                                  ;*                                                                          *
12476                                  ;*  auxnum has printer number                                               *
12477                                  ;*                                                                          *
12478                                  ;****************************************************************************
12479
12480                                  prn_writ:                              ; 2C7h:21Fh = 70h:278Fh
12481                                          ;jcxz   short prn_done ; no chars to output
12482 0000021E E3FC                            jcxz    prn_done        ; 19/10/2022
12483                                  prn_loop:
12484 00000220 BB0200                          mov     bx, 2           ; retry count
12485                                  prn_out:
12486 00000223 E83600                          call    prnstat         ; get status
12487 00000226 751D                            jnz     short TestPrnError
12488 00000228 268A05                          mov     al, [es:di]     ; get character       to print
12489 0000022B 30E4                            xor     ah, ah
12490 0000022D E82E00                          call    prnop           ; print to printer
12491 00000230 7419                            jz      short prn_con   ; no error - continue
12492 00000232 80FCFF                          cmp     ah, 0FFh        ; MODE_CTRLBRK
12493 00000235 7509                            jnz     short _prnwf
12494 00000237 B00C                            mov     al, 0Ch         ; error_I24_gen_failure
12495 00000239 C606[0C00]00                    mov     byte [altah], 0
12496 0000023E EB08                            jmp     short pmessg
12497                                  ; ---------------------------------------------------------------------------
12498
12499                                  _prnwf:
12500 00000240 F6C401                          test    ah, 1           ; timeoutstatus
12501 00000243 7406                            jz      short prn_con
12502                                  TestPrnError:
12503 00000245 4B                              dec     bx              ; retry until count is exhausted.
12504 00000246 75DB                            jnz     short prn_out
12505                                  pmessg:
12506 00000248 E98CFE                          jmp     bc_err_cnt
12507                                  ; ---------------------------------------------------------------------------
12508
12509                                  prn_con:
12510 0000024B 47                              inc     di              ; point to next char and continue
12511 0000024C E2D2                            loop    prn_loop
12512                                  ;prn_done:
12513                                          ; 12/12/2022
12514                                  prn_done2:
12515                                          ;clc
12516                                          ; cf=0
12517 0000024E C3                              retn
12518                                  ; ---------------------------------------------------------------------------
12519
12520                                  ;****************************************************************************
12521                                  ;*                                                                          *
12522                                  ;*  prn_stat - device driver entry to return printer status                 *
12523                                  ;*                                                                          *
12524                                  ;****************************************************************************
12525
12526                                  prn_stat:                              ; 2C7h:251h = 70h:27C1h
```

```
12527 0000024F E80A00                         call    prnstat         ; device in dx
12528 00000252 75F4                           jnz     short pmessg
12529 00000254 F6C480                         test    ah, 80h         ; notbusystatus
12530                                         ;jnz    short prn_done
12531                                         ; 12/12/2022
12532 00000257 75F5                           jnz     short prn_done2 ; cf=0
12533 00000259 E96BFF                         jmp     z_bus_exit
12534                                 ; ---------------------------------------------------------------------------
12535
12536                                 ;*************************************************************************
12537                                 ;*                                                                       *
12538                                 ;*  prnstat - utility function to call ROM BIOS to check                 *
12539                                 ;*           printer status. Return meaningful error code                *
12540                                 ;*                                                                       *
12541                                 ;*************************************************************************
12542
12543                                 prnstat:
12544 0000025C B402                           mov     ah, 2           ; set command for get status
12545                                                                 ; PRINTER - GET       STATUS
12546                                                                 ; DX = printer port (0-3)
12547                                                                 ; Return: AH = status
12548
12549                                 ; =============== S U B   R O U TI N E =======================================
12550
12551                                 ;*************************************************************************
12552                                 ;*                                                                       *
12553                                 ;*  prnop - call ROM BIOS printer function in ah                         *
12554                                 ;*          return zero true if no error                                 *
12555                                 ;*          return zero false if error, al = error code                  *
12556                                 ;*                                                                       *
12557                                 ;*************************************************************************
12558
12559                                 prnop:
12560                                           ; 20/12/2023 - Retro DOS v5.0
12561                                           ; PCDOS 7.1 IBMBIO.COM
12562
12563                                           ;mov    dx, [auxnum]   ; get printer number
12564                                           ;int    17h
12565
12566 0000025E 1E                              push    ds
12567 0000025F FF36[2100]                     push    word [auxnum]
12568 00000263 31D2                           xor     dx, dx ; 0
12569 00000265 8EDA                           mov     ds, dx
12570 00000267 5A                             pop     dx
12571 00000268 9C                             pushf                   ; simulate int 17h
12572 00000269 FA                             cli
12573                                         ;call   dword ptr ds:5ghghCh
12574 0000026A FF1E5C00                       call    far [17h*4]     ; 0:5Ch = INT 17h vector
12575 0000026E 1F                             pop     ds
12576
12577                                     ; This check was added to see if this is a case of no
12578                                     ; printer being installed. This tests checks to be sure
12579                                     ; the error is noprinter (30h)
12580
12581 0000026F 50                              push    ax
12582 00000270 80E430                         and     ah, 30h
12583 00000273 80FC30                         cmp     ah, 30h         ; noprinter
12584 00000276 58                             pop     ax
12585 00000277 7506                           jnz     short NextTest
12586 00000279 80E4DF                         and     ah, 0DFh        ; ~nopaperstatus
12587 0000027C 80CC08                         or      ah, 8           ; ioerrstatus
12588
12589                                 ; examine the status bits to see if an error occurred. unfortunately, several
12590                                 ; of the bits are set so we have to pick and choose. we must be extremely
12591                                 ; careful about breaking basic.
12592
12593                                 NextTest:
12594 0000027F F6C428                         test    ah, 28h         ; (ioerrstatus+nopaperstatus)
12595                                                                 ; i/o error?
12596 00000282 740A                           jz      short checknotready ; no, try not ready
12597
12598                                 ; at this point, we know we have an error. the converse is not true
12599
12600 00000284 B009                           mov     al, 9           ; error_I24_out_of_paper
12601                                                                 ; first, assume       out of paper
12602 00000286 F6C420                         test    ah, 20h         ; out of paper set?
12603 00000289 7502                           jnz     short ret1      ; yes, error is       set
12604 0000028B FEC0                           inc     al              ; return al=10 (i/o error)
12605                                 ret1:
12606 0000028D C3                             retn
12607                                 ; ---------------------------------------------------------------------------
12608
12609                                 checknotready:
12610 0000028E B002                           mov     al, 2           ; assume not-ready
12611 00000290 F6C401                         test    ah, 1
12612 00000293 C3                             retn
12613
12614                                 ; ---------------------------------------------------------------------------
12615
12616                                 ;*************************************************************************
12617                                 ;*                                                                       *
12618                                 ;*  prn_tilbusy - output until busy. Used by print spooler.              *
12619                                 ;*                this entry point should never block waiting for        *
12620                                 ;*                device to come ready.                                  *
12621                                 ;*                                                                       *
12622                                 ;*  inputs: cx = count, es:di -> buffer                                  *
12623                                 ;*  outputs: set the number of bytes transferred in the                  *
12624                                 ;*           device driver request packet                                *
12625                                 ;*                                                                       *
12626                                 ;*************************************************************************
12627
12628                                           ; 19/10/2022
12629                                 prn_tilbusy:                            ; 2C7h:28Bh = 70h:27FBh
12630 00000294 89FE                           mov     si, di          ; everything is       set for lodsb
12631                                 prn_tilbloop:
12632 00000296 51                             push    cx
12633 00000297 53                             push    bx
12634 00000298 30FF                           xor     bh, bh
12635 0000029A 8A1E[8004]                     mov     bl, [printdev]
12636 0000029E D1E3                           shl     bx, 1
12637                                         ;mov    cx, ds:wait_count[bx] ;       wait count times to come ready
12638 000002A0 8B8F[8104]                     mov     cx, [wait_count+bx]
12639 000002A4 5B                             pop     bx
12640                                 prn_getstat:
12641 000002A5 E8B4FF                         call    prnstat         ; get status
12642 000002A8 7514                           jnz     short prn_bperr       ; error
12643 000002AA F6C480                         test    ah, 80h         ; ready yet?
12644 000002AD E1F6                           loope   prn_getstat     ; no, go for more
12645 000002AF 59                             pop     cx              ; get original count
12646 000002B0 740D                           jz      short prn_berr ; still not ready => done
12647 000002B2 26                             es
12648 000002B3 AC                             lodsb
12649                                         ;lods   byte ptr es:[si] ; es
12650                                                                 ; lodsb
```

```
12651 000002B4 30E4                             xor     ah, ah
12652 000002B6 E8A5FF                           call    prnop
12653 000002B9 7504                             jnz     short prn_berr ; error
12654 000002BB E2D9                             loop    prn_tilbloop
12655                                           ; 12/12/2022
12656                                           ; cf=0 (prnop)
12657                                           ;clc                    ; normal no-error return
12658 000002BD C3                               retn                    ;   from device driver
12659
12660                                   ; ---------------------------------------------------------------------------
12661
12662                                   prn_bperr:
12663 000002BE 59                               pop     cx              ; restore transfer count from stack
12664                                   prn_berr:
12665 000002BF E915FE                           jmp     bc_err_cnt
12666                                   ; ---------------------------------------------------------------------------
12667
12668                                   ; 15/10/2022
12669
12670                                   ;****************************************************************************
12671                                   ;*                                                                          *
12672                                   ;*  prn_genioctl - get/set printer retry count                              *
12673                                   ;*                                                                          *
12674                                   ;****************************************************************************
12675
12676                                   ; IOCTL.INC (MSDOS 6.0, 1991)
12677                                   ; 11/01/2019
12678
12679                                   ;*********************************;*
12680                                   ; CHARACTER DEVICES (PRINTERS)    ;*
12681                                   ;*********************************;*
12682
12683                                   ;;RAWIO SUB-FUNCTIONS
12684                                   ;;get_retry_count equ 65h
12685                                   ;;set_retry_count equ 45h
12686
12687                                   ;;struc A_RETRYCOUNT
12688                                   ;;.rc_count: resw 1
12689                                   ;;endstruc
12690
12691                                   ;ioc_pc equ 5
12692
12693                                   ; ---------------------------------------------------------------------------
12694
12695                                           ; 19/10/2022
12696                                   prn_genioctl:                           ; 2C7h:2BAh = 70h:282Ah
12697 000002C2 C43E[1200]                       les     di, [ptrsav]
12698 000002C6 26807D0D05                       cmp     byte [es:di+13], 5 ; [es:di+IOCTL_REQ.MAJORFUNCTION]
12699                                                                   ; ioc_pc
12700 000002CB 7403                             jz      short prnfunc_ok
12701
12702                                   prnfuncerr:
12703 000002CD E905FE                           jmp     bc_cmderr
12704                                   ; ---------------------------------------------------------------------------
12705
12706                                   prnfunc_ok:
12707 000002D0 268A450E                         mov     al, [es:di+14] ; [es:di+IOCTL_REQ.MINORFUNCTION]
12708 000002D4 26C47D13                         les     di, [es:di+19] ; [es:di+IOCTL_REQ.GENERICIOCTL_PACKET]
12709 000002D8 30FF                             xor     bh, bh
12710                                           ;mov    bl, ds:printdev         ; get index into retry counts
12711 000002DA 8A1E[8004]                       mov     bl, [printdev]
12712 000002DE D1E3                             shl     bx, 1
12713                                           ;mov    cx, ds:wait_count[bx] ;       pull out retry count for device
12714 000002E0 8B8F[8104]                       mov     cx, [wait_count+bx]
12715 000002E4 3C65                             cmp     al, 65h         ; get_retry_count
12716 000002E6 7407                             jz      short prngetcount
12717 000002E8 3C45                             cmp     al, 45h         ; set_retry_count
12718 000002EA 75E1                             jnz     short prnfuncerr
12719 000002EC 268B0D                           mov     cx, [es:di]
12720                                   prngetcount:
12721                                           ;mov    ds:wait_count[bx], cx
12722 000002EF 898F[8104]                       mov     [wait_count+bx], cx
12723 000002F3 26890D                           mov     [es:di], cx     ; [es:di+A_RETRYCOUNT.RC_COUNT]
12724                                                                   ; return current retry count
12725                                           ; 12/12/2022
12726                                           ; cf=0
12727                                           ;clc
12728 000002F6 C3                               retn
12729                                   ; ---------------------------------------------------------------------------
12730
12731                                   ;****************************************************************************
12732                                   ;*                                                                          *
12733                                   ;*  prn_ioctl_query                                                             *
12734                                   ;*                                                                          *
12735                                   ;*  Added for 5.00                                                          *
12736                                   ;****************************************************************************
12737
12738                                   prn_ioctl_query:                        ; 2C7h:2F0h = 70h:2860h
12739 000002F7 C43E[1200]                       les     di, [ptrsav]
12740 000002FB 26807D0D05                       cmp     byte [es:di+13], 5 ; [es:di+IOCTL_REQ.MAJORFUNCTION]
12741                                                                   ; ioc_pc
12742 00000300 750D                             jnz     short prn_query_err
12743 00000302 268A450E                         mov     al, [es:di+14] ; [es:di+IOCTL_REQ.MINORFUNCTION]
12744 00000306 3C65                             cmp     al, 65h         ; GET_RETRY_COUNT
12745 00000308 7404                             jz      short IOCtlSupported
12746 0000030A 3C45                             cmp     al, 45h         ; SET_RETRY_COUNT
12747 0000030C 7501                             jnz     short prn_query_err
12748                                   IOCtlSupported:
12749                                           ; 12/12/2022
12750                                           ; cf=0
12751                                           ;clc
12752 0000030E C3                               retn
12753                                   ; ---------------------------------------------------------------------------
12754
12755                                   prn_query_err:
12756                                           ; 12/12/2022
12757                                           ;stc
12758 0000030F E9C3FD                           jmp     bc_cmderr ; (bc_cmderr sets cf to 1)
12759                                   ; ---------------------------------------------------------------------------
12760
12761                                   ;****************************************************************************
12762                                   ;*                                                                          *
12763                                   ;*  aux port driver code -- "aux" == "com1"                                     *
12764                                   ;*                                                                          *
12765                                   ;*  the device driver entry/dispatch code sets up auxnum to                 *
12766                                   ;*  give the com port number to use (0=com1, 1=com2, 2=com3...)             *
12767                                   ;*                                                                          *
12768                                   ;****************************************************************************
12769
12770                                   ;   values in ah, requesting function of int 14h in rom bios
12771
12772                                   ;auxfunc_send       equ   1       ;transmit
12773                                   ;auxfunc_receive equ      2       ;read
12774                                   ;auxfunc_status     equ   3       ;request status
```

```
12775
12776                           ;   error flags, reported by int 14h, reported in ah:
12777
12778                           ;flag_data_ready equ      01h     ;data ready
12779                           ;flag_overrun       equ   02h     ;overrun error
12780                           ;flag_parity        equ   04h     ;parity error
12781                           ;flag_frame  equ   08h     ;framing error
12782                           ;flag_break  equ   10h     ;break detect
12783                           ;flag_tranhol_emp equ     20h     ;transmit holding register empty
12784                           ;flag_timeout       equ   80h     ;timeout
12785
12786                           ;   these flags reported in al:
12787
12788                           ;flag_cts    equ   10h     ;clear to send
12789                           ;flag_dsr    equ   20h     ;data set ready
12790                           ;flag_rec_sig       equ   80h     ;receive line signal detect
12791
12792                           ; ---------------------------------------------------------------------------
12793
12794                           ;****************************************************************************
12795                           ;*                                                                          *
12796                           ;*  aux_read - read cx bytes from [auxnum] aux port to buffer               *
12797                           ;*             at es:di                                                     *
12798                           ;*                                                                          *
12799                           ;****************************************************************************
12800
12801                           aux_read:                         ; 2C7h:30Dh = 70h:287Dh
12802                                       ;jcxz   short exvec2
12803 00000312 E311                        jcxz    exvec2          ; 19/10/2022
12804 00000314 E88000                      call    getbx           ; put address of auxbuf      in bx
12805 00000317 30C0                        xor     al, al
12806 00000319 8607                        xchg    al, [bx]
12807 0000031B 08C0                        or      al, al
12808 0000031D 7503                        jnz     short aux2
12809                           aux1:
12810 0000031F E80500                      call    auxin           ; get character       from port
12811                                                              ; won't return if error
12812                           aux2:
12813 00000322 AA                          stosb
12814 00000323 E2FA                        loop    aux1            ; if more characters, go around      again
12815                           exvec2:
12816 00000325 F8                          clc                     ; all done, successful exit
12817                           auxin_retn: ; 18/12/2022
12818 00000326 C3                          retn
12819                           ; ---------------------------------------------------------------------------
12820
12821                           ;****************************************************************************
12822                           ;*                                                                          *
12823                           ;*  auxin - call rom bios to read character from aux port                   *
12824                           ;*          if error occurs, map the error and return one                   *
12825                           ;*          level up to device driver exit code, setting        *
12826                           ;*          the number of bytes transferred appropriately                   *
12827                           ;*                                                                          *
12828                           ;****************************************************************************
12829
12830                           auxin:
12831 00000327 B402                        mov     ah, 2           ; auxfunc_receive
12832 00000329 E83A00                      call    auxop
12833 0000032C F6C40E                      test    ah, 0Eh         ; flag_frame|flag_parity|flag_overrun
12834                                       ;jnz    short arbad     ; skip if any error bits set
12835                                       ;retn
12836                                       ; 25/06/2023 (BugFix)
12837 0000032F 74F5                        jz      short auxin_retn
12838                           ; ---------------------------------------------------------------------------
12839
12840                           arbad:
12841 00000331 58                          pop     ax              ; remove return       address (near call)
12842                                       ;xor    al, al
12843                                       ;or     al, 0B0h        ; flag_rec_sig|       flag_dsr|flag_cts
12844                                       ; 11/08/2023
12845 00000332 B0B0                        mov     al, 0B0h        ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0334h
12846 00000334 E9A0FD                      jmp     bc_err_cnt
12847
12848                           ; ---------------------------------------------------------------------------
12849
12850                           ;****************************************************************************
12851                           ;*                                                                          *
12852                           ;*  aux_rdnd - non-destructive aux port read                                *
12853                           ;*                                                                          *
12854                           ;****************************************************************************
12855
12856                           aux_rdnd:                         ; 2C7h:335h = 70h:28A5h
12857 00000337 E85D00                      call    getbx
12858 0000033A 8A07                        mov     al, [bx]        ; have bx point       to auxbuf
12859 0000033C 08C0                        or      al, al          ; if al is non-zero (char in buffer)
12860 0000033E 7511                        jnz     short auxdrx    ; then return character
12861 00000340 E82100                      call    auxstat         ; if not, get status of      aux device
12862 00000343 F6C401                      test    ah, 1           ; flag_data_ready - test data ready
12863 00000346 740C                        jz      short auxbus    ; then device is busy (not ready)
12864 00000348 A820                        test    al, 20h         ; flag_dsr - test data set ready
12865 0000034A 7408                        jz      short auxbus    ; then device is busy (not ready)
12866 0000034C E8D8FF                      call    auxin           ; else aux is ready, get character
12867 0000034F 8807                        mov     [bx], al
12868                           auxdrx:
12869 00000351 E99EFE                      jmp     rdexit          ; return busy status
12870                           ; ---------------------------------------------------------------------------
12871
12872                           auxbus:
12873 00000354 E970FE                      jmp     z_bus_exit
12874                           ; ---------------------------------------------------------------------------
12875
12876                           ;****************************************************************************
12877                           ;*                                                                          *
12878                           ;*  aux_wrst - return aux port write status                                 *
12879                           ;*                                                                          *
12880                           ;****************************************************************************
12881
12882                           aux_wrst:                         ; 2C7h:355h = 70h:28C5h
12883 00000357 E80A00                      call    auxstat         ; get status of       aux in ax
12884 0000035A A820                        test    al, 20h         ; test data set       ready
12885 0000035C 74F6                        jz      short auxbus    ; then device is busy (not ready)
12886 0000035E F6C420                      test    ah, 20h         ; flag_tranhol_emp - test transmit hold     reg empty
12887 00000361 74F1                        jz      short auxbus    ; then device is busy (not ready)
12888                                       ; 12/12/2022
12889                                       ; cf=0  ; (test instruction resets cf)
12890                                       ;clc
12891 00000363 C3                          retn
12892                           ; ---------------------------------------------------------------------------
12893
12894                           ;****************************************************************************
12895                           ;*                                                                          *
12896                           ;*  auxstat - call rom bios to determine aux port status                    *
12897                           ;*                                                                          *
12898                           ;*  exit:  ax = status                                                      *
```

```
12899                                   ;*          dx = [auxnum]                                       *
12900                                   ;*                                                              *
12901                                   ;******************************************************************************
12902                                   
12903                                   auxstat:
12904 00000364 B403                               mov     ah, 3           ; auxfunc_status
12905                                   
12906                                             ; fall into auxop
12907                                   
12908                                   ; ============== S U B   R O U T I N E =======================================
12909                                   
12910                                   ;******************************************************************************
12911                                   ;*                                                                            *
12912                                   ;*  auxop - perform rom-biox aux port interrupt                               *
12913                                   ;*                                                                            *
12914                                   ;*  entry:  ah = int 14h function number                                      *
12915                                   ;*  exit:   ax = results                                                      *
12916                                   ;*          dx = [auxnum]                                                     *
12917                                   ;*                                                                            *
12918                                   ;******************************************************************************
12919                                   
12920                                   auxop:              ; proc near
12921                                             ; 20/12/2023 - Retro DOS v5.0
12922                                             ;mov    dx, [auxnum]   ; ah=function code
12923                                             ;                      ; 0=init, 1=send, 2=receive, 3=status
12924                                             ;                      ; get port number
12925                                             ;
12926                                             ;int    14h            ; SERIAL I/O - GET USART STATUS
12927                                             ;                      ; DX = port number (0-3)
12928                                             ;                      ; Return: AX = port status code
12929                                             ; (PCDOS 7.1 IBMBIO.COM)
12930 00000366 1E                               push    ds
12931 00000367 FF36[2100]                       push    word [auxnum]
12932 0000036B 31D2                             xor     dx, dx ; 0
12933 0000036D 8EDA                             mov     ds, dx
12934 0000036F 5A                               pop     dx
12935 00000370 9C                               pushf                   ; simulate INT 14h
12936 00000371 FA                               cli
12937                                           ;call   dword ptr ds:50h
12938 00000372 FF1E5000                         call    far [14h*4]     ; INT 14h vector (14h*4 = 50h)
12939 00000376 1F                               pop     ds
12940 00000377 C3                               retn
12941                                   
12942                                   ; ---------------------------------------------------------------------------
12943                                   
12944                                   ;******************************************************************************
12945                                   ;*                                                                            *
12946                                   ;*  aux_flsh - flush aux input buffer - set contents of                       *
12947                                   ;*             auxbuf [auxnum] to zero                                        *
12948                                   ;*                                                                            *
12949                                   ;*  cas - shouldn't this code call the rom bios input function                *
12950                                   ;*        repeatedly until it isn't ready?  to flush out any                  *
12951                                   ;*        pending serial input queue if there's a tsr like MODE               *
12952                                   ;*        which is providing interrupt-buffering of aux port?                 *
12953                                   ;*                                                                            *
12954                                   ;******************************************************************************
12955                                   
12956                                   aux_flsh:                       ; 2C7h:36Ch = 70h:28DCh
12957 00000378 E81C00                             call    getbx         ; flush aux input buffer
12958 0000037B C60700                             mov     byte [bx], 0  ; get bx to point to auxbuf
12959                                                                   ; zero out buffer
12960                                             ;clc                  ; all done, successful return
12961                                             ; 12/12/2022
12962                                             ; cf=0 ('add' instruction in 'getbx')
12963 0000037E C3                               retn
12964                                   ; ---------------------------------------------------------------------------
12965                                   
12966                                   ;******************************************************************************
12967                                   ;*                                                                            *
12968                                   ;*  aux_writ - write to aux device                                                *
12969                                   ;*                                                                            *
12970                                   ;******************************************************************************
12971                                   
12972                                   aux_writ:                       ; 2C7h:374h = 70h:28E4h
12973                                             ;jcxz   short exvec2  ; write to aux device (if cx > 0)
12974 0000037F E3A4                             jcxz    exvec2        ; 19/10/2022
12975                                   aux_loop:
12976 00000381 268A05                           mov     al, [es:di]   ; get character       to be written
12977                                                                 ; move di pointer to next character
12978 00000384 47                               inc     di
12979 00000385 B401                             mov     ah, 1         ; auxfunc_send - indicates a write
12980 00000387 E8DCFF                           call    auxop         ; send character over aux port
12981 0000038A F6C480                           test    ah, 80h       ; check for error
12982 0000038D 7405                             jz      short awok    ; then no error
12983 0000038F B00A                             mov     al, 10        ; else indicate       write fault
12984 00000391 E943FD                           jmp     bc_err_cnt    ; call error routines
12985                                   ; ---------------------------------------------------------------------------
12986                                   
12987                                   awok:
12988 00000394 E2EB                             loop    aux_loop      ; if cx is non-zero,
12989                                                                 ; still more character to print
12990                                             ;clc                ; all done, successful return
12991                                             ; 12/12/2022
12992                                             ; cf=0 (test instruction above)
12993 00000396 C3                               retn
12994                                   
12995                                   ; ============== S U B   R O U T I N E =======================================
12996                                   
12997                                   ;******************************************************************************
12998                                   ;*                                                                            *
12999                                   ;*  getbx - return bx -> single byte input buffer for                         *
13000                                   ;*          selected aux port ([auxnum])                                      *
13001                                   ;*                                                                            *
13002                                   ;******************************************************************************
13003                                   
13004                                   getbx:
13005 00000397 8B1E[2100]                       mov     bx, [auxnum]  ; return bx -> single byte input buffer
13006                                                                 ; for selected aux port      ([auxnum])
13007                                             ;add    bx, offset auxbuf
13008 0000039B 81C3[1600]                       add     bx, auxbuf    ; 19/10/2022
13009                                             ; 12/12/2022
13010                                             ; cf=0 (if [auxnum] is valid number)
13011 0000039F C3                               retn
13012                                   
13013                                   ; ---------------------------------------------------------------------------
13014                                   
13015                                   ; 15/10/2022
13016                                   
13017                                   ;--------------------------------------------------------------
13018                                   ;                                                             :
13019                                   ;              clock device driver                            :
13020                                   ;                                                             :
13021                                   ;                                                             :
13022                                   ;   this file contains the clock device driver.             :
```

```
13023                                   ;                                                       :
13024                                   ;   the routines in this files are:                             :
13025                                   ;                                                       :
13026                                   ;   routine               function                    :
13027                                   ;   -------               --------                    :
13028                                   ;   tim_writ              set the current time        :
13029                                   ;   tim_read              read the current time       :
13030                                   ;   time_to_ticks         convert time to corresponding:
13031                                   ;                             number of clock ticks     :
13032                                   ;                                                       :
13033                                   ; the clock ticks at the rate of:                       :
13034                                   ;                                                       :
13035                                   ;   1193180/65536 ticks/second (about 18.2 ticks per second):
13036                                   ; see each routine for information on the use.                  :
13037                                   ;                                                       :
13038                                   ;-------------------------------------------------------------
13039
13040                                   ; convert time to ticks
13041                                   ; input : time in cx and dx
13042                                   ; ticks returned in cx:dx
13043
13044                                   ;19/07/2019
13045                                   ;09/03/2019
13046
13047                                   time_to_ticks:                          ; 0070h:2906h =       02C7h:0396h
13048
13049                                   ; first convert from hour,min,sec,hund. to
13050                                   ; total number of 100th of seconds
13051
13052 000003A0 B03C                                mov     al, 60
13053 000003A2 F6E5                                mul     ch              ; hours to minutes
13054 000003A4 B500                                mov     ch, 0
13055 000003A6 01C8                                add     ax, cx          ; total minutes
13056 000003A8 B97017                              mov     cx, 6000        ; 60*100
13057 000003AB 89D3                                mov     bx, dx          ; get out of the way of      the multiply
13058 000003AD F7E1                                mul     cx              ; convert to 1/100 sec
13059 000003AF 89C1                                mov     cx, ax
13060 000003B1 B064                                mov     al, 100
13061 000003B3 F6E7                                mul     bh              ; convert seconds to 1/100 sec
13062 000003B5 01C1                                add     cx, ax          ; combine seconds with hours and min
13063 000003B7 83D200                              adc     dx, 0           ; ripple carry
13064 000003BA B700                                mov     bh, 0
13065 000003BC 01D9                                add     cx, bx          ; combine 1/100       sec
13066 000003BE 83D200                              adc     dx, 0
13067
13068                                       ; dx:cx is time in 1/100 sec
13069
13070 000003C1 92                                  xchg    ax, dx
13071 000003C2 91                                  xchg    ax, cx          ; now time is in cx:ax
13072 000003C3 BB0BE9                              mov     bx, 59659
13073 000003C6 F7E3                                mul     bx              ; multiply low half
13074 000003C8 87CA                                xchg    dx, cx
13075 000003CA 92                                  xchg    ax, dx          ; cx->ax, ax->dx, dx->cx
13076 000003CB F7E3                                mul     bx              ; multiply high       half
13077 000003CD 01C8                                add     ax, cx          ; combine overlapping products
13078 000003CF 83D200                              adc     dx, 0
13079 000003D2 92                                  xchg    ax, dx          ; ax:dx=time*59659
13080 000003D3 BB0500                              mov     bx, 5
13081 000003D6 F6F3                                div     bl              ; divide high half by 5
13082 000003D8 88C1                                mov     cl, al
13083 000003DA B500                                mov     ch, 0
13084 000003DC 88E0                                mov     al, ah          ; remainder of divide-by-5
13085 000003DE 98                                  cbw
13086 000003DF 92                                  xchg    ax, dx          ; use it to extend low half
13087 000003E0 F7F3                                div     bx              ; divide low half by 5
13088 000003E2 89C2                                mov     dx, ax          ; cx:dx is now number of ticks in time
13089 000003E4 CB                                  retf                    ; far return
13090
13091                                   ; ---------------------------------------------------------------------------
13092
13093                                   ; 17/10/2022
13094                                   ; 15/10/2022
13095
13096                                   ;-------------------------------------------------------------
13097                                   ;
13098                                   ; tim_writ sets the current time
13099                                   ;
13100                                   ; on entry es:[di] has the current time:
13101                                   ;
13102                                   ;   number of days since 1-1-80   (word)
13103                                   ;   minutes (0-59)                (byte)
13104                                   ;   hours (0-23)                  (byte)
13105                                   ;   hundredths of seconds (0-99)  (byte)
13106                                   ;   seconds (0-59)                (byte)
13107                                   ;
13108                                   ; each number has been checked for the correct range.
13109                                   ;
13110                                   ;   NOTE: Any changes in this routine probably require corresponding
13111                                   ;   changes in the version that is built with the power manager driver.
13112                                   ;   See ptime.asm.
13113                                   ;
13114                                   ;-------------------------------------------------------------
13115
13116                                       ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
13117                                       ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:03EAh
13118                                   tim_writ:                               ; 2C7h:3DBh = 70h:294Bh
13119 000003E5 268B05                             mov     ax, [es:di]
13120 000003E8 50                                  push    ax              ; daycnt. we need to set this at the very
13121                                                                      ; end to avoid tick windows.
13122 000003E9 803E[8C04]00                        cmp     byte [havecmosclock], 0
13123                                              ;cmp    ds:havecmosclock, 0
13124 000003EE 7423                                jz      short no_cmos_1
13125 000003F0 268A4503                            mov     al, [es:di+3]  ; near indirect       calls
13126                                                                      ; get binary hours
13127                                                                      ; convert to bcd
13128                                              ;call   far [bintobcd]
13129                                              ;;call  ds:bintobcd    ; call far [bintobcd]
13130                                              ; 08/08/2023
13131 000003F4 E8E800                              call    bintobcd
13132 000003F7 88C5                                mov     ch, al         ; ch = bcd hours
13133 000003F9 268A4502                            mov     al, [es:di+2]  ; get binary minutes
13134                                              ;call   far [bintobcd]
13135                                              ;;call  ds:bintobcd    ; convert to bcd
13136 000003FD E8DF00                              call    bintobcd
13137 00000400 88C1                                mov     cl, al         ; cl = bcd minutes
13138 00000402 268A4505                            mov     al, [es:di+5]  ; get binary seconds
13139                                              ;call   far [bintobcd]
13140                                              ;;call  ds:bintobcd
13141 00000406 E8D600                              call    bintobcd
13142
13143 00000409 88C6                                mov     dh, al         ; dh = bcd seconds
13144 0000040B B200                                mov     dl, 0          ; dl = 0 (st) or 1 (dst)
13145 0000040D FA                                  cli
13146 0000040E B403                                mov     ah, 3
```

```
13147 00000410 CD1A                             int     1Ah             ; CLOCK - SET REAL TIME        CLOCK (AT,XT286,CONV,PS)
13148                                           ; CH = hours in        BCD, CL = minutes in BCD
13149                                           ;  DH = seconds in BCD,DL = 01h        if daylight savings, 00h if standard
time
13150                                           ; Return: CMOS clock set
13151 00000412 FB                               sti
13152                                   no_cmos_1:
13153 00000413 268B4D02                         mov     cx, [es:di+2]
13154 00000417 268B5504                         mov     dx, [es:di+4]
13155                                           ; 17/10/2022
13156 0000041B FF1E[0606]                       call    far [ttticks]
13157                                           ;call   dword ptr ds:ttticks ; call far      [ttticks]
13158                                           ; convert time to ticks
13159                                           ; cx:dx now has time in ticks
13160 0000041F FA                               cli                     ; turn off timer
13161 00000420 B401                             mov     ah, 1
13162 00000422 CD1A                             int     1Ah             ; CLOCK - SET TIME OF DAY
13163                                           ; CX:DX = clock count
13164                                           ; Return: time of day set
13165                                           ;pop    ds:daycnt
13166 00000424 8F06[8904]                       pop     word [daycnt]
13167 00000428 FB                               sti
13168                                           ;cmp    ds:havecmosclock, 0
13169 00000429 803E[8C04]00                     cmp     byte [havecmosclock], 0
13170 0000042E 7409                             jz      short no_cmos_2
13171
13172                                           ; 08/08/2023
13173                                           ;call   far [daycnttoday]
13174                                           ;;call  ds:daycnttoday ; call far [daycnttoday]
13175                                           ; convert to bcd format
13176 00000430 E80700                           call    daycnttoday
13177
13178 00000433 FA                               cli
13179 00000434 B405                             mov     ah, 5
13180 00000436 CD1A                             int     1Ah             ; CLOCK - SET DATE IN REAL TIME       CLOCK (AT,XT286,CONV,PS)
13181                                           ; DL = day in BCD, DH =       month in BCD, CL = year       in BCD
13182                                           ; CH = century (19h or 20h)
13183                                           ; Return: CMOS clock set
13184 00000438 FB                               sti
13185                                   no_cmos_2:
13186                                           ; 12/12/2022
13187                                           ; cf=0
13188                                           ;clc
13189 00000439 C3                               retn
13190
13191                                   ; ---------------------------------------------------------------------------
13192
13193                                   ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
13194                                   ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0440h
13195                                   %if 1
13196
13197                                   ; CMOS Clock setting support routines used by MSCLOCK.
13198                                   ; Warning!!! This code will be dynamically relocated by MSINIT.
13199
13200                                   daycnttoday:       ; proc near
13201
13202                                   ; entry: [daycnt] = number of days since 1-1-80
13203                                   ;
13204                                   ; return: ch - century in bcd
13205                                   ;     cl - year in bcd
13206                                   ;     dh - month in bcd
13207                                   ;     dl - day in bcd
13208
13209                                           ; 20/12/2023 - Retro DOS v5.0
13210
13211                                           ; 08/08/2023 (ds:) (near proc)
13212                                           ; 16/10/2022 (cs:) (far proc)
13213 0000043A FF36[8904]                       push    word [daycnt] ; save daycnt
13214 0000043E 813E[8904]891C                   cmp     word [daycnt], 7305 ; (365*20+(20/4))
13215                                                                   ; # days from 1-1-1980 to 1-1-2000
13216 00000444 7308                             jnb     short century20
13217                                           ;mov    byte [base_century], 19
13218                                           ;mov    byte [base_year], 80
13219                                           ; 08/08/2023
13220 00000446 C706[8D04]1350                   mov     word [base_century], 5013h
13221 0000044C EB0C                             jmp     short years
13222                                   ; ---------------------------------------------------------------------------
13223
13224                                   century20:
13225                                           ;mov    byte [base_century], 20
13226                                           ;mov    byte [base_year], 0
13227                                           ; 08/08/2023
13228 0000044E C706[8D04]1400                   mov     word [base_century], 20
13229 00000454 812E[8904]891C                   sub     word [daycnt], 7305 ; (365*20+(20/4))
13230                                                                   ; adjust daycnt
13231                                   years:
13232 0000045A 31D2                             xor     dx, dx
13233 0000045C A1[8904]                         mov     ax, [daycnt]
13234 0000045F BBB505                           mov     bx, 1461        ; (366+365*3)
13235                                           ; # of days in a Leap year block
13236 00000462 F7F3                             div     bx              ; AX = # of leap block,       DX = daycnt
13237 00000464 8916[8904]                       mov     [daycnt], dx    ; save daycnt left
13238 00000468 B304                             mov     bl, 4
13239 0000046A F6E3                             mul     bl              ; AX = # of years. Less       than 100
13240 0000046C 0006[8E04]                       add     [base_year], al ; So, ah = 0. Adjust year
13241 00000470 FF06[8904]                       inc     word [daycnt]  ; set daycnt to        1 base
13242                                           ; 08/08/2023
13243 00000474 BB6E01                           mov     bx, 366
13244 00000477 B90300                           mov     cx, 3
13245                                           ;cmp    word [daycnt], 366 ; daycnt=remainder of leap year
13246 0000047A 391E[8904]                       cmp     [daycnt], bx ; 366
13247 0000047E 7619                             jbe     short leapyear ; within 366+355+355+355 days.
13248 00000480 FE06[8E04]                       inc     byte [base_year] ; if daycnt <= 366, then leap year
13249                                           ;sub    word [daycnt], 366 ; else daycnt--, base_year++ ;
13250 00000484 291E[8904]                       sub     [daycnt], bx ; 366 ; 08/08/2023
13251                                           ;mov    cx, 3           ; And next three years are normal
13252                                           ; 08/08/2023
13253 00000488 4B                               dec     bx ; 365
13254                                   regularyear:       ; 20/12/2023
13255                                           ;cmp    word [daycnt], 365 ; for(i=1; i>3 or daycnt <=365; i++)
13256 00000489 391E[8904]                       cmp     [daycnt], bx ; 365 ; 08/08/2023
13257 0000048D 760F                             jbe     short yeardone ; {if (daycnt >        365)
13258 0000048F FE06[8E04]                       inc     byte [base_year] ; { daycnt -=        365
13259                                           ;sub    word [daycnt], 365 ; }
13260 00000493 291E[8904]                       sub     [daycnt], bx ; 365 ; 08/08/2023
13261 00000497 E2F0                             loop    regularyear    ; }
13262                                                                   ;
13263                                                                   ; should never fall through loop
13264                                   leapyear:
13265 00000499 C606[9004]1D                     mov     byte [february], 29 ; 08/08/2023
13266                                           ;mov    byte [month_tab+1], 29 ; leap year.
13267                                                                   ; change month table.
13268                                   yeardone:
13269 0000049E 31DB                             xor     bx, bx
```

```
000004A0 31D2                           xor     dx, dx
000004A2 A1[8904]                       mov     ax, [daycnt]
                                        ;mov    si, offset month_tab
000004A5 BE[8F04]                       mov     si, month_tab  ; 19/10/2022
                                        ;mov    cx, 12
                                        ; 08/08/2023
000004A8 B10C                           mov     cl, 12
                                months:
000004AA FEC3                           inc     bl
                                        ; 08/08/2023
000004AC 8A14                           mov     dl, [si]        ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:04B7h
000004AE 39D0                           cmp     ax, dx          ; cmp daycnt for each month till fit
                                                                ; dh=0
000004B0 7605                           jbe     short month_done
000004B2 46                             inc     si              ; next month
000004B3 29D0                           sub     ax, dx          ; adjust daycnt
000004B5 E2F3                           loop    months          ;
                                                                ; should never fall through loop
                                month_done:
000004B7 C606[9004]1C                   mov     byte [february], 28 ; 08/08/2023
                                        ;mov    byte [month_tab+1], 28
                                                                ; restore month table value
000004BC 88DA                           mov     dl, bl
000004BE 8A36[8E04]                     mov     dh, [base_year]
000004C2 8A0E[8D04]                     mov     cl, [base_century] ; al=day,dl=month,dh=year,cl=cntry
000004C6 E81600                         call    bintobcd        ; convert "day"       to bcd
                                                                ; dl = bcd day,       al = month
000004C9 86C2                           xchg    dl, al
000004CB E81100                         call    bintobcd        ; dh = bcd month, al = year
000004CE 86C6                           xchg    dh, al
000004D0 E80C00                         call    bintobcd        ; cl = bcd year, al = century
000004D3 86C1                           xchg    cl, al
000004D5 E80700                         call    bintobcd        ; ch = bcd century
000004D8 88C5                           mov     ch, al
000004DA 8F06[8904]                     pop     word [daycnt] ; restore original value
000004DE C3                             retn

                                ;---------------------------------------------------------------------------
                                bintobcd:   ; proc near           ; real time clock support

                                ;convert a binary input in al (less than 63h or 99 decimal)
                                ;into a bcd value in al. ah destroyed.

000004DF D40A                           aam                     ; AH = AL/10, AL = AL MOD 10
000004E1 D510                           aad     10h               ; db 0D5h,10h
                                                                ; AL = (AH*10H)+AL, AH = 0
000004E3 C3                             retn
                                %endif

                                ;---------------------------------------------------------------------------

                                ; 15/10/2022

                                ;---------------------------------------------------------------------------
                                ; gettime reads date and time
                                ; and returns the following information:
                                ;
                                ;   es:[di]  =count of days since 1-1-80
                                ;   es:[di+2]=hours
                                ;   es:[di+3]=minutes
                                ;   es:[di+4]=seconds
                                ;   es:[di+5]=hundredths of seconds
                                ;
                                ;   NOTE: Any changes in this routine probably require corresponding
                                ;   changes in the version that is built with the power manager driver.
                                ;   See ptime.asm.
                                ;---------------------------------------------------------------------------

                                tim_read:                       ; 2C7h:435h = 70h:29A5h
000004E4 E84A00                         call    GetTickCnt
000004E7 8B36[8904]                     mov     si, [daycnt]

                                ; we now need to convert the time in tick to the time in 100th of
                                ; seconds. the relation between tick and seconds is:
                                ;
                                ;          65,536 seconds
                                ;         ----------------
                                ;         1,193,180 tick
                                ;
                                ; to get to 100th of second we need to multiply by 100. the equation is:
                                ;
                                ;   ticks from clock  * 65,536 * 100
                                ;      --------------------------------- = time in 100th of seconds
                                ;           1,193,180
                                ;
                                ; fortunately this formula simplifies to:
                                ;
                                ;   ticks from clock * 5 * 65,536
                                ;      --------------------------------- = time in 100th of seconds
                                ;           59,659
                                ;
                                ; the calculation is done by first multipling tick by 5. next we divide by
                                ; 59,659. in this division we multiply by 65,536 by shifting the dividend
                                ; my 16 bits to the left.
                                ;
                                ; start with ticks in cx:dx
                                ; multiply by 5

000004EB 89C8                           mov     ax, cx
000004ED 89D3                           mov     bx, dx          ; start with ticks in cx:dx
                                                                ; multiply by 5
000004EF D1E2                           shl     dx, 1
000004F1 D1D1                           rcl     cx, 1           ; times 2
000004F3 D1E2                           shl     dx, 1
000004F5 D1D1                           rcl     cx, 1           ; times 4
000004F7 01DA                           add     dx, bx
000004F9 11C8                           adc     ax, cx          ; times 5
000004FB 92                             xchg    ax, dx

                                ; now have ticks * 5 in   dx:ax
                                ; we now need to multiply by 65536 and divide by 59659 d.

000004FC B9OBE9                         mov     cx, 59659       ; get divisor
000004FF F7F1                           div     cx              ; dx now has remainder
                                                                ; ax has high word of final quotient

                                        ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                        ;mov    bx, ax          ; put high word       in safe place
00000501 93                             xchg    bx, ax
00000502 31C0                           xor     ax, ax          ; this is the multiply by 65536
00000504 F7F1                           div     cx              ; bx:ax now has time in 100th of seconds

                                ; rounding based on the   remainder may be added here
```

```
                                ; the result in    bx:ax is time in 1/100 second.
                                
00000506 89DA                             mov     dx, bx
00000508 B9C800                           mov     cx, 200         ; extract 1/100's
                                
                                ; division by 200 is necessary to ensure no overflow--max result
                                ; is number of seconds in a day/2 = 43200.
                                
0000050B F7F1                             div     cx
0000050D 80FA64                           cmp     dl, 100         ; remainder over 100?
00000510 7203                             jb      short noadj
00000512 80EA64                           sub     dl, 100         ; keep 1/100's less than 100
                                noadj:
00000515 F5                               cmc                     ; if we subtracted 100,       carry is now set
00000516 88D3                             mov     bl, dl          ; save 1/100's
                                
                                ; to compensate    for dividing by       200 instead of 100, we now multiply
                                ; by two, shifting a one in if the remainder had exceeded 100.
                                
00000518 D1D0                             rcl     ax, 1
0000051A B200                             mov     dl, 0
0000051C D1D2                             rcl     dx, 1
                                          ;mov    cx, 60          ; divide out seconds
                                          ; 20/12/2023
0000051E B13C                             mov     cl, 60
00000520 F7F1                             div     cx
00000522 88D7                             mov     bh, dl          ; save the seconds
00000524 F6F1                             div     cl              ; break into hours and minutes
00000526 86E0                             xchg    al, ah
                                
                                ; time is now in ax:bx (hours, minutes, seconds, 1/100 sec)
                                
                                          ; 08/08/2023
                                          ;push   ax
                                          ;mov    ax, si          ; daycnt
00000528 96                               xchg    ax, si
00000529 AB                               stosw
                                          ;pop    ax
0000052A 96                               xchg    ax, si          ; al = hours, ah = minutes
0000052B AB                               stosw
0000052C 89D8                             mov     ax, bx
0000052E AB                               stosw
0000052F F8                               clc                     ; [es:di] = count of days since 1-1-80
                                                                  ;   [es:di+2] = hours
                                                                  ;   [es:di+3] = minutes
                                                                  ;   [es:di+4] = seconds
                                                                  ;   [es:di+5] = hundredths of seconds
00000530 C3                               retn
                                
                                ; =============== S U B   R O U T I N E =======================================
                                
                                ; 15/10/2022
                                
                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : GetTickCnt
                                ;
                                ;          Returns the tick count in CX:DX. Takes care of DayCnt in case
                                ;          of rollover [except when power management driver is in use].
                                ;          Uses the following logic for updating Daycnt
                                ;
                                ;          if ( rollover ) {
                                ;                  if ( t_switch )
                                ;                          daycnt++ ;
                                ;                  else
                                ;                          daycnt += rollover ;
                                ;          }
                                ;
                                ; USES : AX
                                ;
                                ; RETURNS : CX:DX - tick count
                                ; MODIFIES : daycnt
                                ;
                                ;--------------------------------------------------------------------------
                                
                                          ; 17/10/2022
                                GetTickCnt:
00000531 30E4                             xor     ah, ah
00000533 CD1A                             int     1Ah             ; CLOCK - GET TIME OF DAY
                                                                  ; Return: CX:DX       = clock count
                                                                  ; AL = 00h if clock was       read or written (via AH=0,1) since the previous
                                                                  ; midnight
                                                                  ; Otherwise, AL       > 0
                                          ; 20/12/2023
00000535 30E4                             xor     ah, ah
00000537 3826[8B04]                       cmp     byte [t_switch], ah ; 0
                                          ;cmp    byte [t_switch], 0 ; use old method ? (>0 is yes)
0000053B 7505                             jnz     short inc_case ; old method assumes that Int 1Ah returns rollover flag
                                          ;xor    ah, ah          ; new method assumes that Int 1Ah returns roll over count
                                                                  ; and not flag
0000053D 0106[8904]                       add     [daycnt], ax
00000541 C3                               retn
                                ; ---------------------------------------------------------------------------
                                
                                inc_case:
00000542 08C0                             or      al, al
00000544 7404                             jz      short no_rollover
00000546 FF06[8904]                       inc     word [daycnt]
                                no_rollover:
0000054A C3                               retn
                                
                                ; ---------------------------------------------------------------------------
                                ; ---------------------------------------------------------------------------
                                ; 03/10/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
                                ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0556h
                                
                                %if 1
                                
0000054B 4641543132202020       fat_12_id:  db 'FAT12   '
00000553 4641543136202020       fat_16_id:  db 'FAT16   '
0000055B 4641543332202020       fat_32_id:  db 'FAT32   '
00000563 4E4F204E414D452020-    nul_vid:    db 'NO NAME    '
0000056C 2020
                                
                                %endif
                                
                                ;--------------------------------------------------------------------------
                                ; MSDISK.ASM - MSDOS 6.0 - 1991
                                ;--------------------------------------------------------------------------
                                ; 15/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ; 09/03/2019 - Retro DOS v4.0
                                
                                ; MSDISK.ASM - MSDOS 3.3 - 02/02/1988
                                ; 26/05/2018 - Retro DOS v3.0
```

```
                        ; 23/03/2018 - Retro DOS v2.0

                        ;error_unknown_media equ  7      ; for use in BUILD BPB call

                        ;struc BPB_TYPE
                        ;.SECSIZE:  resw 1
                        ;.SECALL:   resb 1
                        ;.RESNUM:   resw 1
                        ;.FATNUM:   resb 1
                        ;.DIRNUM:   resw 1
                        ;.SECNUM:   resw 1
                        ;.FATID:    resb 1
                        ;.FATSIZE:  resw 1
                        ;.SLIM:            resw 1
                        ;.HLIM:            resw 1
                        ;.HIDDEN:   resw 1
                        ;.size:
                        ;endstruc

                        ;--------------------------------------------------------------------
                        ;   disk interface routines
                        ;--------------------------------------------------------------------

                        ; device attribute bits:
                        ;   bit 6 - get/set map for logical drives and generic ioctl.

                        ;MAXERR            equ    5
                        ;MAX_HD_FMT_ERR    equ    2

                        ;LSTDRV     equ 504h

                        ; some floppies do not have changeline. as a result, media-check would
                        ; normally return i-don't-know, the dos would continually reread the fat and
                        ; discard cached data. we optimize this by implementing a logical door-latch:
                        ; it is physically impossible to change a disk in under 2 seconds. we retain
                        ; the time of the last successful disk operation and compare it with the current
                        ; time during media-check. if < 2 seconds and at least 1 timer tick has passed,
                        ; the we say no change. if > 2 seconds then we say i-don't-know. finally,
                        ; since we cannot trust the timer to be always available, we record the number
                        ; of media checks that have occurred when no apparent time has elapsed. while
                        ; this number is < a given threshold, we say no change. when it exceeds that
                        ; threshold, we say i-don't-know and reset the counter to 0. when we store
                        ; the time of last successful access, if we see that time has passed too,
                        ; we reset the counter.

                        accessmax   equ    5

                        ; due to various bogosities, we need to continually adjust what the head
                        ; settle time is.  the following algorithm is used:
                        ;
                        ;   get the current head settle value.
                        ;   if it is 0, then
                        ;   set slow = 15
                        ;   else
                        ;   set slow = value
                        ;   ...
                        ;*****************************************
                        ;************ old algorithm ***************
                        ;*   if we are seeking and writing then
                        ;*   use slow
                        ;*   else
                        ;*   use fast
                        ;*****************************************
                        ;*********** ibm's requested logic ***********
                        ;   if we are seeking and writing and not on an at then
                        ;   use slow
                        ;   else
                        ;   use fast
                        ;   ...
                        ;   restore current head settle value
                        ;
                        ;
                        ;---------------------------------------
                        multrk_on   equ    10000000b      ;user spcified mutitrack=on, or system turns
                                                          ; it on after handling config.sys file as a
                                                          ; default value, if multrk_flag = multrk_off1.
                        multrk_off1 equ    00000000b      ;initial value. no "multitrack=" command entered.
                        multrk_off2 equ    00000001b      ;user specified multitrack=off.

                        ; close data segment, open Bios_Code segment

                        ; 15/10/2022

                        ; BIOSCODE:04A2h (MSDOS 6.21, IO.SYS)

                        ;--------------------------------------------------------------------
                        ;   command jump table
                        ;--------------------------------------------------------------------

0000056E 00                             db 0

                                        ; 11/12/2022
                                        %if 0

                                        dsktbl:            db 26                ; 2C7h:4A2h = 70h:2A12h
                                                                              ; ((dtbl_siz-1)/2) ; this is the size of the table ; 26
                                                  dw 1742h                    ; dsk_init
                                                  dw 4EBh                     ; media_chk
                                                  dw 592h                     ; get_bpb
                                                  dw 0D5h                     ; bc_cmderr
                                                  dw 857h                     ; dsk_read
                                                  dw 83Dh                     ; x_bus_exit
                                                  dw 558h                     ; ret_carry_clear
                                                  dw 558h                     ; ret_carry_clear
                                                  dw 849h                     ; dsk_writ
                                                  dw 841h                     ; dsk_writv
                                                  dw 558h                     ; ret_carry_clear
                                                  dw 558h                     ; ret_carry_clear
                                                  dw 0D5h                     ; bc_cmderr
                                                  dw 80Ah                     ; dsk_open
                                                  dw 81Ah                     ; dsk_close
                                                  dw 831h                     ; dsk_rem
                                                  dw 558h                     ; ret_carry_clear
                                                  dw 558h                     ; ret_carry_clear
                                                  dw 558h                     ; ret_carry_clear
                                                  dw 0C6Bh                    ; do_generic_ioctl
                                                  dw 558h                     ; ret_carry_clear
                                                  dw 558h                     ; ret_carry_clear
                                                  dw 558h                     ; ret_carry_clear
                                                  dw 1124h                    ; ioctl_getown
                                                  dw 1142h                    ; ioctl_setown
                                                  dw 129Ah                    ; ioctl_support_query

                        ;dtbl_siz equ $-dsktbl
```

```
13641
13642                                   %endif
13643
13644                                   ; 21/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
13645                                   ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0579h
13646
13647                                           ; 21/12/2023 - Retro DOS v5.0
13648                                           ; 11/12/2022
13649 0000056F 1A                       dsktbl:         db (dtbl_siz-1)/2     ; 26 ; this is the size of the table
13650 00000570 [4A1A]                           dw dsk_init
13651 00000572 [B805]                           dw media_chk
13652 00000574 [5706]                           dw get_bpb
13653                                           ;dw bc_cmderr
13654 00000576 [4A0E]                           dw ioctl_input ; PCDOS 7 ; 21/12/2023
13655 00000578 [7209]                           dw dsk_read
13656 0000057A [5809]                           dw x_bus_exit
13657 0000057C [2206]                           dw ret_carry_clear
13658 0000057E [2206]                           dw ret_carry_clear
13659 00000580 [6409]                           dw dsk_writ
13660 00000582 [5C09]                           dw dsk_writv
13661 00000584 [2206]                           dw ret_carry_clear
13662 00000586 [2206]                           dw ret_carry_clear
13663                                           ;dw bc_cmderr
13664 00000588 [F70D]                           dw ioctl_output ; PCDOS 7 ; 21/12/2023
13665 0000058A [2909]                           dw dsk_open
13666 0000058C [3809]                           dw dsk_close
13667 0000058E [4E09]                           dw dsk_rem
13668 00000590 [2206]                           dw ret_carry_clear
13669 00000592 [2206]                           dw ret_carry_clear
13670 00000594 [2206]                           dw ret_carry_clear
13671 00000596 [CC0E]                           dw do_generic_ioctl
13672 00000598 [2206]                           dw ret_carry_clear
13673 0000059A [2206]                           dw ret_carry_clear
13674 0000059C [2206]                           dw ret_carry_clear
13675 0000059E [A713]                           dw ioctl_getown
13676 000005A0 [C413]                           dw ioctl_setown
13677 000005A2 [1C15]                           dw ioctl_support_query
13678
13679                                   dtbl_siz equ $-dsktbl
13680
13681                                   ; =============== S U B   R O U T I N E =======================================
13682
13683                                   ; ---------------------------------------------------------------------------
13684                                   ; setdrive scans through the data structure of bdss, and returns a pointer to
13685                                   ; the one that belongs to the drive specified. carry is set if none exists
13686                                   ; for the drive. Pointer is returned in es:[di]
13687                                   ;
13688                                   ;  AL contains the logical drive number.
13689                                   ; ---------------------------------------------------------------------------
13690
13691                                   SetDrive:
13692                                           ;les   di, dword ptr ds:start_bds ; Point es:di to first bds
13693 000005A4 C43E[1901]                       les    di, [start_bds] ; 19/10/2022
13694                                   X_Scan_Loop:
13695 000005A8 26384505                         cmp    [es:di+5], al
13696 000005AC 7409                             jz     short X_SetDrv
13697 000005AE 26C43D                           les    di, [es:di]    ; [es:di+BDS.link] ; Go      to next bds
13698 000005B1 83FFFF                           cmp    di, 0FFFFh
13699 000005B4 75F2                             jnz    short X_Scan_Loop
13700 000005B6 F9                               stc
13701                                   X_SetDrv:
13702 000005B7 C3                               retn
13703
13704                                   ; ---------------------------------------------------------------------------
13705
13706                                   ; 15/10/2022
13707
13708                                       ; 21/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
13709                                       ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:05C2h
13710
13711                                   ; ---------------------------------------------------------------------
13712                                   ; if id is f9, have a 96tpi disk else
13713                                   ; if bit 2 is 0 then media is not removable and could not have changed
13714                                   ;  otherwise if within 2 secs of last disk operation media could not
13715                                   ;    have changed, otherwise dont know if media has changed
13716                                   ; ---------------------------------------------------------------------
13717
13718                                   media_chk:                      ; 2C7h:4EBh = 70h:2A5Bh
13719 000005B8 E8E9FF                           call   SetDrive
13720 000005BB BE0100                           mov    si, 1
13721                                           ; 21/12/2023
13722 000005BE 26F6454001                       test   byte [es:di+40h], 1
13723                                           ;test  byte [es:di+24h], 1 ; [es:di+BDS.flags+1]
13724                                                                   ; fchanged_by_format
13725 000005C3 7415                             jz     short WeAreNotFakingIt
13726                                           ; 21/12/2023
13727 000005C5 26806540FE                       and    byte [es:di+40h], 0FEh
13728                                           ; 12/12/2022
13729                                           ;and   byte [es:di+24h], 0FEh ; ~fchanged_by_format
13730                                           ;;and  word [es:di+23h], 0FEFFh ; [es:di+BDS.flags]
13731                                                                   ; ~fchanged_by_format ;      reset flag
13732 000005CA C606[1E01]FF                    mov    byte [tim_drv], 0FFh ; -1
13733                                                                   ; Ensure that we ask the rom if media has changed
13734                                           ; 21/12/2023
13735 000005CF 26F6453F01                       test   byte [es:di+3Fh], 1
13736                                           ;test  byte [es:di+23h], 1 ; [es:di+BDS.flags]
13737                                                                   ; fnon_removable
13738 000005D4 740B                             jz     short wehaveafloppy
13739                                           ;mov   si, 0FFFFh     ; Indicate media changed
13740                                           ; 11/08/2023
13741 000005D6 F7DE                             neg    si             ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:05E0h
13742 000005D8 EB2B                             jmp    short Media_Done ; Media_Done
13743                                   ; ---------------------------------------------------------------------------
13744
13745                                   WeAreNotFakingIt:
13746                                           ; 21/12/2023
13747 000005DA 26F6453F01                       test   byte [es:di+3Fh], 1
13748                                           ;test  byte [es:di+BDS.flags], fnon_removable
13749                                           ;test  byte [es:di+23h], 1
13750 000005DF 7524                             jnz    short Media_Done
13751                                   wehaveafloppy:
13752                                           ;xor   si, si ; 0     ; Presume "I don't know"
13753                                           ; 11/08/2023
13754 000005E1 4E                               dec    si ; 0         ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:05EBh
13755
13756                                           ; If we have a floppy with changeline support, we ask the ROM
13757                                           ; to determine if media has changed. We do not perform the
13758                                           ; 2 second check for these drives.
13759
13760 000005E2 803E[7700]00                     cmp    byte [fhave96], 0     ; Do we have changeline       support?
13761 000005E7 740A                             jz     short mChk_NoChangeLine       ; Brif not
13762 000005E9 E83615                           call   mediacheck     ;  Call into removable routine
13763 000005EC 7236                             jb     short err_exitj
13764 000005EE E89A16                           call   haschange
```

```
13765 000005F1 7512                     jnz     short Media_Done
13766                                   mChk_NoChangeLine:
13767                                   ; If we come here, we have a floppy with no changeline support
13768
13769 000005F3 BE0100                   mov     si, 1           ; Presume no change
13770 000005F6 A0[1E01]                 mov     al, [tim_drv]   ; Last drive accessed
13771 000005F9 263A4504                 cmp     al, [es:di+4]   ; [es:di+BDS.drivenum]
13772                                                           ; Is drive of last access the same?
13773 000005FD 7505                     jnz     short Media_Unk       ; No, then "i don't know"
13774 000005FF E82800                   call    Check_Time_Of_Access
13775 00000602 EB01                     jmp     short Media_Done
13776                                   ; ---------------------------------------------------------------------------
13777
13778                                   Media_Unk:
13779 00000604 4E                       dec     si              ; 0 ; Return "I don't know"
13780
13781                                   ; SI now contains the correct value for media change.
13782                                   ; Clean up the left overs
13783                                   Media_Done:
13784                                   ; 19/10/2022
13785 00000605 06                       push    es
13786 00000606 C41E[1200]               les     bx, [ptrsav]
13787 0000060A 2689770E                 mov     [es:bx+0Eh], si       ; [es:bx+trans]
13788 0000060E 07                       pop     es
13789 0000060F 09F6                     or      si, si
13790 00000611 790F                     jns     short ret_carry_clear ;       volidok
13791 00000613 803E[7700]00             cmp     byte [fhave96], 0
13792 00000618 7403                     jz      short mChk1_NoChangeLine ; Brif      no changeline support
13793 0000061A E80016                   call    media_set_vid
13794                                   mChk1_NoChangeLine:
13795 0000061D C606[1E01]FF             mov     byte [tim_drv], 0FFh ; -1
13796                                                           ; Make sure we ask rom for media check
13797                                   ret_carry_clear:
13798 00000622 F8                       clc                     ; volidok
13799 00000623 C3                       retn
13800                                   ; ---------------------------------------------------------------------------
13801
13802                                   err_exitj:
13803 00000624 E88207                   call    maperror        ; guaranteed to       set carry
13804                                   ret81:
13805 00000627 B481                     mov     ah, 81h         ; return error status
13806 00000629 C3                       retn                    ; return with carry set
13807
13808                                   ; =============== S U B   R O U T I N E =======================================
13809
13810                                   ; ---------------------------------------------------------------------------
13811                                   ; perform a check on the time passed since the last access for this physical
13812                                   ; drive.
13813                                   ; we are accessing the same drive. if the time of last successful access was
13814                                   ; less than 2 seconds ago, then we may presume that the disk was not changed.
13815                                   ; returns in si:
13816                                   ;   0 - if time of last access was >= 2 seconds
13817                                   ;   1 - if time was < 2 seconds (i.e no media change assumed)
13818                                   ; registers affected ax,cx,dx, flags.
13819                                   ;
13820                                   ;   assume es:di -> bds, ds->Bios_Data
13821                                   ; ---------------------------------------------------------------------------
13822
13823                                               ; 21/12/2023 - Retro DOS v5.0 IBMBIO.COM
13824                                               ; 19/10/2022
13825                                   Check_Time_Of_Access:
13826 0000062A BE0100                   mov     si, 1           ; presume no change.
13827 0000062D E801FF                   call    GetTickCnt      ; cx:dx is the elapsed time
13828                                               ; 21/12/2023
13829 00000630 268B4579                 mov     ax, [es:di+79h]
13830                                   ;mov    ax, [es:di+47h]       ; [es:di+BDS.tim_lo]
13831                                                           ; get stored time
13832 00000634 29C2                     sub     dx, ax
13833                                               ; 21/12/2023
13834 00000636 268B457B                 mov     ax, [es:di+7Bh]
13835                                   ;mov    ax, [es:di+49h]       ; [es:di+BDS.tim_hi]
13836 0000063A 19C1                     sbb     cx, ax
13837                                               ; 11/08/2023
13838                                               ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0646h
13839                                   ;mov    al, [accesscount]
13840 0000063C 7515                     jnz     short timecheck_unk ; cx<>0 => >1 hour
13841 0000063E 09D2                     or      dx, dx          ; time must pass
13842 00000640 750C                     jnz     short timepassed ; yes, examine max value
13843                                               ; 11/08/2023
13844                                   ;inc    al
13845                                   ;cmp    al, 5
13846                                   ;;inc   byte [accesscount]
13847                                   ;;cmp   byte [accesscount], 5
13848                                                           ; if count is less than threshold, ok
13849                                   ;jb     short timecheck_ret
13850                                   ;;dec   byte [accesscount] ; don't let the count wrap
13851                                               ; 11/08/2023
13852                                   ;dec    al
13853                                   ;jmp    short timecheck_unk ; "i don't know" if media changed
13854                                               ; 11/08/2023
13855 00000642 803E[1D01]04                     cmp     byte [accesscount], 4
13856 00000647 730A                     jnb     short timecheck_unk
13857 00000649 FE06[1D01]               inc     byte [accesscount]
13858 0000064D C3                       retn
13859
13860                                   ; ---------------------------------------------------------------------------
13861
13862                                   timepassed:
13863 0000064E 83FA24                   cmp     dx, 36          ; 18*2 ; 18.2 tics per second.
13864                                                           ; min elapsed time? (2 seconds)
13865 00000651 7601                     jbe     short timecheck_ret ; yes, presume no change
13866
13867                                               ; everything indicates that we do not know what has happened.
13868                                   timecheck_unk:
13869 00000653 4E                       dec     si              ; presume i don't know
13870                                   timecheck_ret:
13871                                               ; 11/08/2023
13872                                               ;mov    [accesscount], al
13873 00000654 C3                       retn
13874
13875                                   ; ---------------------------------------------------------------------------
13876                                   ; 15/10/2022
13877                                   Err_Exitj2:
13878 00000655 EBCD                     jmp     short err_exitj
13879
13880                                   ; ---------------------------------------------------------------------------
13881
13882                                   ; 15/10/2022
13883
13884                                   ; ===========================================================================
13885                                   ; Build a valid bpb for the disk in the drive.
13886                                   ; ===========================================================================
13887
13888                                               ; 21/12/2023 - Retro DOS v5.0 IBMBIO.COM
```

```
                                        ; 19/10/2022
                                get_bpb:                                ; 2C7h:592h = 70h:2B02h
00000657 268A25                         mov     ah, [es:di]     ; get fat id byte read by dos
0000065A E847FF                         call    SetDrive        ; get the correct bds for the drive
                                        ; 21/12/2023
0000065D 26F6453F01                     test    byte [es:di+3Fh], 1
                                        ;test   byte [es:di+23h], 1 ; [es:di+BDS.flags]
                                                                ; fnon_removable
00000662 7523                           jnz     short already_gotbpb ; no need to build     for fixed disks

                                        ; let's set the default value for volid,vol_serial,
                                        ; filesys_id in bds table

00000664 E83600                         call    clear_ids
                                        ;mov    ds:set_id_flag,       1 ; indicate to       set system id in bds
00000667 C606[9B04]01                   mov     byte [set_id_flag], 1
0000066C E86700                         call    GetBp           ; build a bpb if necessary
0000066F 72B6                           jb      short ret81
                                        ;cmp    ds:set_id_flag,       2 ; already, volume_label set from boot
00000671 803E[9B04]02                   cmp     byte [set_id_flag], 2
                                        ;mov    ds:set_id_flag,       0 ; record to bds table?
00000676 C606[9B04]00                   mov     byte [set_id_flag], 0
0000067B 740A                           jz      short already_gotbpb ; do not set it again from     root dir
                                                                ; otherwise, conventional boot record
                                        ;cmp    ds:fhave96, 0  ; do we have changeline       support?
0000067D 803E[7700]00                   cmp     byte [fhave96], 0
00000682 7403                           jz      short already_gotbpb ; brif not
00000684 E80A16                         call    set_volume_id
                                already_gotbpb:
00000687 83C706                         add     di, 6           ; BDS.BPB
                                                                ; return the bpb from the current bds

                                ;               fall into setptrsav, es:di -> result

                                ; ---------------------------------------------------------------------------

                                ; 15/10/2022

                                ; ===========================================================================
                                ;Setptrsav is also jumped to from dsk_init (msbio2.asm). In both cases, the
                                ;pointer to be returned is in es:di. We were incorrectly returning ds:di.
                                ;Note that this works in most cases because most pointers are in Bios_Data.
                                ;It fails, for instance, when we install an external drive using driver.sys
                                ;because then the BDS segment is no longer Bios_Data.
                                ;NB: It is fine to corrupt cx because this is not a return value and anyway
                                ;this returns to Chardev_entry (msbio1.asm) where all registers are
                                ;restored before returning to the caller.
                                ; ===========================================================================

                                ; 21/12/2023
                                %if 0
                                        ; 19/10/2022
                                SetPtrSav:  ; return point for dsk_init
                                        mov     cx, es          ; save es
                                        ;les    bx, ds:ptrsav
                                        les     bx, [ptrsav]
                                        mov     [es:bx+0Dh], ah         ; [es:bx+media]
                                        mov     [es:bx+12h], di         ; [es:bx+count]
                                        mov     [es:bx+14h], cx         ; [es:bx+count+2]
                                        clc
                                        retn
                                %endif
                                        ; 21/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
                                        ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0698h
                                SetPtrSav:
                                        ; return point for dsk_init
0000068A 1E                             push    ds
                                        ;lds    bx, ds:ptrsav
0000068B C51E[1200]                     lds     bx, [ptrsav]
0000068F 88670D                         mov     [bx+0Dh], ah   ; [bx+media]
00000692 897F12                         mov     [bx+12h], di   ; [bx+count]
00000695 8C4714                         mov     [bx+14h], es   ; [bx+count+2]
00000698 1E                             push    ds
00000699 07                             pop     es
0000069A 1F                             pop     ds
0000069B F8                             clc
0000069C C3                             retn

                                ; =============== S U B   R O U T I N E =======================================

                                ; 15/10/2022

                                ; ---------------------------------------------------
                                ; clear ids in bds table. only applied for floppies.
                                ;input:  es:di -> bds table
                                ;   assumes ds: -> Bios_Data
                                ;output: volid set to "NO NAME    "
                                ;    vol_serial set to 0.
                                ;    filesys_id set to "FAT12   " or "FAT16   "
                                ;      depending on the flag fatsize in bds.
                                ;
                                ;   trashes si, cx
                                ; ---------------------------------------------------

                                ;size_of_EXT_BOOT_VOL_LABEL equ 11
                                ;size_of_EXT_SYSTEM_ID equ 8

                                ; 11/09/2023
                                ; 14/08/2023
                                ;BDS.fatsiz equ 1Fh
                                ; 21/12/2023
                                ;BDS.fatsiz equ 59

                                        ; 22/12/2023
                                        ; 21/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
                                clear_ids:
                                        ;mov    al, [es:di+1Fh] ; mov al,[es:di+BDS.fatsiz]
                                        ; 21/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM BugFix)
0000069D 268A5D3B                       mov     bl, [es:di+3Bh] ; mov bl,[es:di+BDS.fatsiz]; *+
                                clear_ids_x:
                                        ; 21/12/2023
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:06ABh)
                                        ; 11/09/2023
                                        ; (MSDOS 5.0 IO.SYS - BIOSCODE:05D9h)
000006A1 57                             push    di
000006A2 31C9                           xor     cx, cx          ; no serial number
                                        ; 21/12/2023
000006A4 26898D8900                     mov     [es:di+89h], cx         ; [es:di+BDS.vol_serial]
000006A9 26898D8B00                     mov     [es:di+8Bh], cx         ; [es:di+BDS.vol_serial+2]
                                        ;mov    [es:di+57h], cx         ; [es:di+BDS.vol_serial]
                                        ;mov    [es:di+59h], cx         ; [es:di+BDS.vol_serial+2]

                                        ; BUGBUG - there's a lot in common here and with
                                        ; mov_media_ids.. see if we can save some space by
```

```
                                        ; merging them... jgl
                                        
                                        ;mov    cx, 11          ; size_of_EXT_BOOT_VOL_LABEL
                                        ; 10/12/2022
000006AE B10B                           mov     cl, 11 ; cx = 11
                                        
                                        ;;mov   si, offset vol_no_name ; "NO NAME    "
                                        ;mov    si, vol_no_name        ; 19/10/2022
                                        ; 22/12/2023
                                        ;mov    si, offset nul_vid ; "NO NAME    "
000006B0 BE[6305]                       mov     si, nul_vid
                                        
                                        ; 21/12/2023
000006B3 83C77D                         add     di, 125
                                        ;add    di, 75          ; BDS.volid
                                        
                                        ;rep movsb
                                        ; 21/12/2023
                                        ;rep movs byte ptr es:[di], byte ptr cs:[si] ; cs rep movsb
                                        ; 26/12/2023
                                        ;cs     ; vol_no_name is in BIOSCODE segment
                                        ;rep movsb
000006B6 F3                             rep
000006B7 2E                             cs
000006B8 A4                             movsb
                                        
                                        ; 11/09/2023 (BugFix, DI is not start addr of BDS structure here)
                                        ;;test byte [es:di+BDS.fatsiz], fbig
                                        ; (MSDOS 5.0 IO.SYS - BIOSCODE:05EFh)
                                        ;test   byte [es:di+1Fh], 40h
                                        ; 21/12/2023 - Retro DOS v5.0
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:06C3h)
                                        ;test   byte [es:di+59], 20h
                                                ; (here, es:di points to the BDS offset +136)
                                                ; purpose: test byte [es:di+BDS.fatsiz], fbigbig
                                                ; applied: test byte [es:BDS.fatsiz+136], fbigbig -BUG!-
                                                ; (PCDOS 7.1 BUG note: 26/06/2023 - Erdogan Tan)
                                                ;; ! NOTE - 11/08/2023 - Erdogan Tan (Retro DOS v4.2 IO.SYS bugfix)
                                                ; Microsoft/IBM code has a bug here because the BDS's
                                                ; .volid and .filesys_id fields will be reset
                                                ; (to their default text) according to 'BDS.fatsiz' flags
                                                ; at the BDS offset 59 but current (this) code checks flags
                                                ; at ES:DI+59 while DI points the BDS offset 136!? ; (PCDOS 7.1)
                                                ;; at the BDS offset 31 but current (this) code checks flags
                                                ;; at ES:DI+31 while DI points the BDS offset 86!? ; (MSDOS 6.22)
                                                ;
                                                ; Correct Code:
                                                ; ;test byte [ES:59],20h or [ES:BDS.fatsiz],fbigbig  ; (PCDOS 7.1)
                                                ; ;;test byte [ES:31],40h or [ES:BDS.fatsiz],fbig  ; (MSDOS 6.22)
                                                ; 11/09/2023
                                                ; (before 'rep movsb') 'mov al,[es:di+BDS.Fatsiz]' and then
                                                ; (after 'rep movsb') 'test al,fbig' (AL is free/proper to use here)
                                                ;
                                                ; Same BUG is existing in MSDOS 6.22 IO.SYS - BIOSCODE:05EFh
                                                ; and in Windows ME IO.SYS - BIOSCODE:0E1Ah as 'test byte [es:di+59],20h'
                                                ;
                                                ; (Why this bug did not affect MSDOS and PCDOS 7.x applications:
                                                ; 'clear_ids' is used for floppy disks only and the default
                                                ; option of 'clear_ids' is FAT12 volid and filesys_id text
                                                ; when the flag bit has wrong value for FAT16/40h or FAT32/20h.)
                                        
                                        ; 21/12/2023 - Retro DOS v5.0
                                        ;mov    si, offset fat_32_id ; "FAT32   "
000006B9 BE[5B05]                       mov     si, fat_32_id
                                        
                                        ; 21/12/2023
                                        ; BugFix (of the PCDOS 7.1 IBMBIO.COM BUG) ; *+
                                        ;test   bl, fbigbig ; FAT32 flag
000006BC F6C320                         test    bl, 20h ; * ; BL = [es:BDS.fatsiz] = [es:59]
000006BF 750B                           jnz     short ci_bigfat
                                        
                                        ;mov    si, offset fat_16_id ; "FAT16 "
000006C1 BE[5305]                       mov     si, fat_16_id  ; 19/10/2022
                                        
                                        ; 21/12/2023
                                        ; !BUG! (PCDOS 7.1 IBMBIO.COM BIOSCODE:06CDh)
                                        ;test   byte [es:di+59], 40h ; [es:di+BDS.fatsiz], fbig
                                        ; BugFix ; *+
                                        ;test   bl, fbig ; FAT16 flag
000006C4 F6C340                         test    bl, 40h ; * ; Retro DOS v5.0
                                        ;;test  al, 40h ; * ; Retro DOS v4.2
000006C7 7503                           jnz     short ci_bigfat
                                        
                                        ;mov    si, offset fat_12_id ; "FAT12 "
000006C9 BE[4B05]                       mov     si, fat_12_id  ; 19/10/2022
                                        
                                        ci_bigfat:
                                                ;mov    cx, 8           ; size_of_EXT_SYSTEM_ID
                                                ; 10/12/2022
000006CC B108                                   mov     cl, 8 ; cx = 8
000006CE 83C705                                 add     di, 5           ; (BDS.filesys_id-BDS.volid)-size_of_EXT_BOOT_VOL_LABEL
                                                                        ; filesys_id field
                                                ;rep movsb
                                                ; 21/12/2023 - Retro DOS v5.0
                                                ;rep movs byte ptr es:[di], byte ptr cs:[si] ; 0F3h,2Eh,0A4h
                                                ; 26/12/2023
                                                ;cs     ; fat32_id, fat16_id and fat12_id are in BIOSCODE segment
                                                ;rep movsb
000006D1 F3                                     rep
000006D2 2E                                     cs
000006D3 A4                                     movsb
                                        
000006D4 5F                                     pop     di              ; restore bds pointer
                                        getret_exit:                    ; 21/12/2023
000006D5 C3                                     retn
                                        
                                        ; =============== S U B   R O U T I N E =======================================
                                        
                                        ; 15/10/2022
                                        
                                        ; ---------------------------------------------------------------------------
                                        ;   getbp - return bpb from the drive specified by the bds.
                                        ;       if the return_fake_bpb flag is set, then it does nothing.
                                        ;       note that we never come here for fixed disks.
                                        ;       for all other cases,
                                        ;         - it reads boot sector to pull out the bpb
                                        ;         - if no valid bpb is found, it then reads the fat sector,
                                        ;           to get the fat id byte to build the bpb from there.
                                        ;
                                        ;   inputs: es:di point to correct bds.
                                        ;
                                        ;   outputs:       fills in bpb in current bds if valid bpb or fat id on disk.
                                        ;           carry set, and al=7 if invalid disk.
```

```
14137                                   ;           carry set and error code in al if other error.
14138                                   ;           if failed to recognize the boot record, then will set the
14139                                   ;           set_id_flag to 0.
14140                                   ;           this routine will only work for a floppy diskette.
14141                                   ;                for a fixed disk, it will just return.
14142                                   ;
14143                                   ;   ****** Note: getbp is a clone of getbp which uses the newer
14144                                   ;     segment definitions. It should be migrated towards.
14145                                   ;      now es:di has the bds, ds: has Bios_Data
14146                                   ; ---------------------------------------------------------------------------
14147
14148                                             ; 29/12/2023
14149                                             ; 21/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
14150                                   GetBp:
14151                                             ; if returning fake bpb then return bpb as is.
14152                                             ; 21/12/2023
14153 000006D6 26F6453F05                         test    byte [es:di+3Fh], 5 ; PCDOS 7.1
14154                                             ;test   byte [es:di+BDS.flags], return_fake_bpb|fnon_removable
14155                                             ;test   byte [es:di+23h], 5 ; MSDOS 6.22 (& MSDOS 5.0)
14156                                             ;jz     short getbp1   ; getbp1
14157                                             ;jmp    getret_exit
14158                                             ; 21/12/2023
14159 000006DB 75F8                             jnz     short getret_exit
14160                                   ; ---------------------------------------------------------------------------
14161                                   getbp1:
14162 000006DD 51                               push    cx
14163 000006DE 52                               push    dx
14164 000006DF 53                               push    bx
14165
14166                                             ; attempt to read in boot sector and determine bpb.
14167                                             ; we assume that the 2.x and greater dos disks all
14168                                             ; have a valid boot sector.
14169
14170 000006E0 E8CF00                           call    readbootsec
14171 000006E3 720A                             jb      short getbp_err_ret_brdg ; carry set if there was error.
14172 000006E5 09DB                             or      bx, bx          ; bx is 0 if boot sector is valid.
14173 000006E7 7509                             jnz     short dofatbpb
14174 000006E9 E81401                           call    movbpb          ; move bpb into       registers
14175                                           ;jmp    short Has1
14176                                           ; 21/12/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
14177 000006EC E9B500                           jmp     getret
14178                                   ; ---------------------------------------------------------------------------
14179
14180                                   getbp_err_ret_brdg:
14181 000006EF E9B600                           jmp     getbp_err_ret
14182                                   ; ---------------------------------------------------------------------------
14183
14184                                             ; we have a 1.x diskette. In this case read in the fat ID byte
14185                                             ; and fill in bpb from there.
14186                                   dofatbpb:
14187 000006F2 E8B401                           call    readfat         ; puts media descriptor       byte in ah
14188 000006F5 72F8                             jb      short getbp_err_ret_brdg
14189                                           ;cmp    ds:fhave96, 0  ;  changeline support available?
14190 000006F7 803E[7700]00                     cmp     byte [fhave96], 0 ; 19/10/2022
14191 000006FC 7403                             jz      short bpb_nochangeline ; brif not
14192 000006FE E83115                           call    hidensity       ; may not return! May add sp, 2       and
14193                                                                   ; jump to has1!!!!!! or       has720K
14194                                   bpb_nochangeline:         ; test for a valid 3.5" medium
14195                                             ; 21/12/2023 - Retro DOS v5.0
14196 00000701 26807D3E02                       cmp     byte [es:di+3Eh], 2
14197                                           ;cmp    byte [es:di+22h], 2 ; [es:di+BDS.formfactor]
14198                                                                   ; ffSmall
14199 00000706 7512                             jnz     short is_floppy
14200 00000708 80FCF9                           cmp     ah, 0F9h        ; is it a valid fat id byte for       3.5" ?
14201 0000070B 7512                             jnz     short got_unknown_medium
14202                                   Has720K:
14203                                             ; 21/12/2023
14204                                             ;;mov   bx, offset sm92 ; pointer to correct bpb
14205                                             ;mov    bx, sm92        ; 19/10/2022
14206
14207                                             ; es points to segment of bds. the following should be modified
14208                                             ; to get spf,csec,spau,spt correctly. it had been wrong if
14209                                             ; driver.sys is loaded since the bds is inside the driver.sys.
14210
14211                                             ; 21/12/2023
14212                                             ;; 10/12/2022
14213                                             ;;mov   al, [bx+0]      ; [bx+bpbtype.spf]
14214                                             ;; 21/12/2022
14215                                             ;mov    al, [bx]
14216                                             ;mov    cx, [bx+3]      ; [bx+bpbtype.csec]
14217                                             ;mov    dx, [bx+5]      ; [bx+bpbtype.spau]
14218                                             ;mov    bx, [bx+1]      ; [bx+bpbtype.spt]
14219                                             ;; 19/10/2022 - Temporary !
14220                                             ;;db    8Ah, 87h, 0, 0 ; mov al, [bx+0]
14221                                             ;;db    8Bh, 8Fh, 3, 0 ; mov cx, [bx+3]
14222                                             ;;db    8Bh, 97h, 5, 0 ; mov dx, [bx+5]
14223                                             ;;db    8Bh, 9Fh, 1, 0 ; mov bx, [bx+1]
14224
14225                                             ; 21/12/2023 - Retro DOS v5.0
14226 0000070D B003                             mov     al, 3           ; bpbtype.sbf = 3
14227 0000070F B9A005                           mov     cx, 1440        ; bpbtype.csec = 1440
14228 00000712 BA0202                           mov     dx, 202h        ; dl = bpbtype.spau = 2
14229                                                                   ; dh = bpbtype.chead = 2
14230 00000715 BB0970                           mov     bx, 7009h       ; bl = bpbtype.spt = 9
14231                                                                   ; bh = bpbtype.dire = 112
14232 00000718 EB30                             jmp     short Has1
14233                                   ; ---------------------------------------------------------------------------
14234
14235                                   is_floppy:                ; must be a 5.25" floppy if we come here
14236 0000071A 80FCF8                           cmp     ah, 0F8h        ; valid media?? (0F8h-0FFh)
14237                                           ;jb     short got_unknown_medium
14238                                             ; 21/12/2023
14239 0000071D 730A                             jnb     short chk_160K
14240                                   ; ---------------------------------------------------------------------------
14241                                             ; 21/12/2023
14242                                             ; we have a 3.5" diskette for which we cannot build a bpb.
14243                                             ; we do not assume any type of bpb for this medium.
14244                                   got_unknown_medium:
14245                                           ;mov    ds:set_id_flag,       0
14246 0000071F C606[9B04]00                     mov     byte [set_id_flag], 0
14247 00000724 B007                             mov     al, 7
14248 00000726 F9                               stc
14249 00000727 EB7B                             jmp     short getret
14250                                   ; ---------------------------------------------------------------------------
14251                                   chk_160K:
14252 00000729 B001                             mov     al, 1           ; set number of       fat sectors
14253 0000072B BB0840                           mov     bx, 16392       ; 64*256+8
14254                                                                   ; set dir entries and sector max
14255 0000072E B94001                           mov     cx, 320         ; 40*8
14256                                                                   ; set size of drive
14257 00000731 BA0101                           mov     dx, 257         ; 01*256+1
14258                                                                   ; set head limit and sec/all unit
14259                                             ; 21/12/2023
14260                                           ;mov    al, 1           ; bpbtype.sbf = 1
```

```
14261                                   ;mov    bx, 4008h       ; bl = bpbtype.spt = 8
14262                                   ;                       ; bh = bpbtype.dire = 64
14263                                   ;mov    cx, 140h        ; bpbtype.csec = 320
14264                                   ;mov    dx, 101h        ; dl = bpbtype.spau = 1
14265                                   ;                       ; dh = bpbtype.chead = 1
14266
14267 00000734 F6C402                   test    ah, 2           ; test for 8 or       9 sector
14268 00000737 7505                     jnz     short has8      ; nz = has 8 sectors
14269
14270                                   ; 29/12/2023
14271                                   ;inc    al      ; 2     ; inc number of       fat sectors
14272                                   ;inc    bl      ; 9     ; inc sector max
14273 00000739 40                       inc     ax
14274 0000073A 43                       inc     bx
14275
14276                                   ;add    cx, 40          ; increase size       (to 360)
14277                                   ; 18/12/2022
14278 0000073B 80C128                   add     cl, 40  ; 28h   ; 180K (360 sectors)
14279                                 has8:
14280 0000073E F6C401                   test    ah, 1           ; test for 1 or       2 heads
14281 00000741 7407                     jz      short Has1      ; jz = 1 head
14282 00000743 01C9                     add     cx, cx          ; double size of disk
14283 00000745 B770                     mov     bh, 112         ; increase number of directory entries
14284 00000747 FEC6                     inc     dh      ; 2     ; inc sec/all unit
14285                                   ; 29/12/2023
14286                                   ;inc    dl      ; 2     ; inc head limit
14287 00000749 42                       inc     dx
14288                                 Has1:
14289                                   ; 02/09/2023 (PCDOS 7.1, IBMBIO.COM - BIOSCODE:0754h)
14290 0000074A 1E                       push    ds
14291 0000074B 06                       push    es
14292 0000074C 1F                       pop     ds
14293
14294                                   ;mov    [es:di+8], dh   ; [es:di+BDS.secperclus]
14295                                   ;mov    [es:di+0Ch], bh         ; [es:di+BDS.direntries]
14296                                   ;mov    [es:di+0Eh], cx         ; [es:di+BDS.totalsecs16]
14297                                   ;mov    [es:di+10h], ah         ; [es:di+BDS.media]
14298                                   ;mov    [es:di+11h], al         ; [es:di+BDS.fatsecs]
14299                                   ;mov    [es:di+13h], bl         ; [es:di+BDS.secpertrack]
14300                                   ;mov    [es:di+15h], dl         ; [es:di+BDS.heads]
14301
14302 0000074D 887508                   mov     [di+8], dh      ; [di+BDS.secperclus]
14303 00000750 30F6                     xor     dh, dh
14304 00000752 895515                   mov     [di+15h], dx    ; [di+BDS.heads]
14305 00000755 88FA                     mov     dl, bh
14306 00000757 89550C                   mov     [di+0Ch], dx    ; [di+BDS.direntries]
14307 0000075A 894D0E                   mov     [di+0Eh], cx    ; [di+BDS.totalsecs16]
14308 0000075D 894D1B                   mov     [di+1Bh], cx    ; [di+BDS.totalsecs32]
14309 00000760 886510                   mov     [di+10h], ah    ; [di+BDS.media]
14310 00000763 88C2                     mov     dl, al
14311 00000765 895511                   mov     [di+11h], dx    ; [di+BDS.fatsecs]
14312 00000768 88DA                     mov     dl, bl
14313 0000076A 895513                   mov     [di+13h], dx    ; [di+BDS.secpertrack]
14314
14315                                   ; the BDS_BPB.BPB_HIDDENSECTORS+2 field and the
14316                                   ; BDS_BPB.BPB_BIGTOTALSECTORS field need to be set
14317                                   ; to 0 since this code is for floppies
14318
14319                                   ; 18/12/2022
14320                                   ;mov    word [es:di+19h], 0 ; [es:di+BDS.hiddensecs+2]
14321                                   ;mov    word [es:di+17h], 0 ; [es:di+BDS.hiddensecs]
14322                                   ;mov    word [es:di+1Dh], 0 ; [es:di+BDS.totalsecs32+2]
14323                                   ; 18/12/2022
14324 0000076D 29C9                     sub     cx, cx ; 0
14325                                   ;mov    [es:di+19h], cx ; 0 ; [es:di+BDS.hiddensecs+2]
14326                                   ;mov    [es:di+17h], cx ; 0 ; [es:di+BDS.hiddensecs]
14327                                   ;mov    [es:di+1Dh], cx ; 0 ; [es:di+BDS.totalsecs32+2]
14328
14329                                   ; 02/09/2023
14330 0000076F 894D19                   mov     [di+19h], cx ; 0 ; [di+BDS.hiddensecs+2]
14331 00000772 894D17                   mov     [di+17h], cx ; 0 ; [di+BDS.hiddensecs]
14332 00000775 894D1D                   mov     [di+1Dh], cx ; 0 ; [di+BDS.totalsecs32+2]
14333
14334                                   ; 21/12/2023 - Retro DOS v5.0
14335 00000778 894D1F                   mov     [di+1Fh], cx    ; [di+BDS.fatsecs32] ; BPB_FATSz32
14336 0000077B 894D21                   mov     [di+21h], cx    ; [di+BDS.fatsecs32+2]
14337 0000077E 894D27                   mov     [di+27h], cx    ; [di+BDS.rootdirclust]
14338 00000781 894D29                   mov     [di+29h], cx    ; [di+BDS.rootdirclust+2]
14339 00000784 894D2F                   mov     [di+2Fh], cx    ; [di+BDS.reserved]
14340                                                           ;     BPB_Reserved (12 zero bytes)
14341 00000787 894D31                   mov     [di+31h], cx
14342 0000078A 894D33                   mov     [di+33h], cx
14343 0000078D 894D35                   mov     [di+35h], cx
14344 00000790 894D37                   mov     [di+37h], cx
14345 00000793 894D39                   mov     [di+39h], cx
14346 00000796 894D23                   mov     [di+23h], cx    ; [di+BDS.extflags] ; BPB_ExtFlags
14347 00000799 894D25                   mov     [di+25h], cx    ; [di+BDS.fsver] ; BPB_FSVer
14348
14349 0000079C 49                       dec     cx              ; -1 ; 0FFFFFFFFh
14350 0000079D 894D2B                   mov     [di+2Bh], cx    ; [di+BDS.fsinfo] ; BPB_FSInfo
14351 000007A0 894D2D                   mov     [di+2Dh], cx    ; [di+BDS.bkbootsec] ; BPB_BkBootSec
14352
14353 000007A3 1F                       pop     ds ; 02/09/2023
14354                                 getret:
14355 000007A4 5B                       pop     bx
14356 000007A5 5A                       pop     dx
14357 000007A6 59                       pop     cx
14358                                 ;getret_exit:             ; 21/12/2023
14359 000007A7 C3                       retn
14360                                 ; ---------------------------------------------------------------------------
14361
14362                                 getbp_err_ret:  ; before doing anything else, set set_id_flag to 0.
14363                                   ;mov    ds:set_id_flag,       0
14364                                   ; 19/10/2022
14365 000007A8 C606[9B04]00             mov     byte [set_id_flag], 0
14366 000007AD E8F905                   call    maperror
14367 000007B0 EBF2                     jmp     short getret
14368                                 ; ---------------------------------------------------------------------------
14369                                 ; 21/12/2023
14370                                   ; we have a 3.5" diskette for which we cannot build a bpb.
14371                                   ; we do not assume any type of bpb for this medium.
14372                                   ;
14373                                 ;got_unknown_medium:
14374                                   ;mov    ds:set_id_flag,       0
14375                                   ;mov    byte [set_id_flag], 0
14376                                   ;mov    al, 7
14377                                   ;stc
14378                                   ;jmp    short getret
14379
14380                                 ; =============== S U B   R O U T I N E =======================================
14381
14382                                 ; 15/10/2022 - Retro DOS 4.0 (Modified MSDOS 5.0 IO.SYS)
14383
14384                                 ; ---------------------------------------------------------------
```

```
                                ; read in the boot sector. set carry if error in reading sector.
                                ; bx is set to 1 if the boot sector is invalid, otherwise it is 0.
                                ;
                                ;   assumes es:di -> bds, ds-> Bios_Data
                                ; ----------------------------------------------------------------

                                ; 10/03/2019 - Retro DOS v4.0

                                ; 30/12/2022 - Retro DOS v4.2
                                ; (MSDOS 6.21 IO.SYS, BIOSCODE:06C3h)
                                ; ((MSDOS 6.22 IO.SYS, BIOSCODE:06C3h)) ; 22/12/2023

                                ; 22/12/2023 - Retro DOS v5.0
                                ; (PCDOS 7.1 IBMBIO.COM, BIOSCODE:07C6h)

                                readbootsec:
000007B2 B600                               mov     dh, 0           ; head 0
000007B4 B90100                             mov     cx, 1           ; cylinder 0, sector 1
000007B7 E8FC00                             call    read_sector
000007BA 7243                               jb      short err_ret
000007BC 31DB                               xor     bx, bx          ; assume valid boot sector

                                            ; put a sanity check for the boot sector in here to detect
                                            ; boot sectors that do not have valid bpbs. we examine the
                                            ; first two bytes - they must contain a long jump (69h) or a
                                            ; short jump (EBh) followed by a nop (90h), or a short jump
                                            ; (E9h). if this test is passed, we further check by examining
                                            ; the signature at the end of the boot sector for the word
                                            ; AA55h. if the signature is not present, we examine the media
                                            ; descriptor byte to see if it is valid. for dos 3.3, this
                                            ; logic is modified a little bit. we are not going to check
                                            ; signature. instead we are going to sanity check the media
                                            ; byte in bpb regardless of the validity of signature. this is
                                            ; to save the already developed commercial products that have
                                            ; good jump instruction and signature but with the false bpb
                                            ; informations

                                ; that will crash the diskette drive operation. (for example, symphony diskette).

                                            ; 02/09/2023
                                            ; 19/10/2022
                                            ;cmp    byte [disksector], 69h ; is it a direct jump?
                                            ;jz     short check_bpb_mediabyte ; don't need to find a nop
                                            ;cmp    byte [disksector], 0E9h ; dos 2.0 jump?
                                            ;jz     short check_bpb_mediabyte ; no need for     nop
                                            ;cmp    byte [disksector], 0EBh ; how about a short jump?
                                            ;jnz    short invalidbootsec
                                            ;cmp    byte [disksector+2], 90h ; is next one a nop?
                                            ;jnz    short invalidbootsec

                                            ; 02/09/2023 (PCDOS 7.1)
000007BE A0[5201]                           mov     al, [disksector]
000007C1 3C69                               cmp     al, 69h         ; is it a direct jump?
000007C3 740F                               je      short check_bpb_mediabyte
                                                                    ; don't need to find a nop
000007C5 3CE9                               cmp     al, 0E9h        ; dos 2.0 jump?
000007C7 740B                               je      short check_bpb_mediabyte
                                                                    ; no need for nop
000007C9 3CEB                               cmp     al, 0EBh        ; how about a short jump?
000007CB 7530                               jne     short invalidbootsec
000007CD 803E[5401]90                       cmp     byte [disksector+2], 90h ; is next one a nop?
000007D2 7529                               jne     short invalidbootsec

                                ; 15/10/5022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ;
                                ;; 10/03/2019
                                ;; (MSDOS 3.3, MSDISK.ASM, 1988)
                                ;;
                                ;; Don't have to perform the following signature check since
                                ;; we need to check the media byte even with the good signatured diskette.
                                ;;
                                ;;check_signature:
                                ;;          cmp     word [cs:disksector+1FEh],0AA55h ; see if non-ibm
                                ;;                                                   ; disk or 1.x media.
                                ;;          jz      short checksinglesided ; go see if singled sided medium.
                                ;;                                         ; may need some special handling

                                ; check for non-ibm disks which do not have the signature AA55h at the
                                ; end of the boot sector, but still have a valid boot sector. this is done
                                ; by examining the media descriptor in the boot sector.

                                            ; 19/10/2022
                                check_bpb_mediabyte:
000007D4 A0[6701]                           mov     al, [disksector+15h]
                                                                    ; [disksector+EXT_BOOT.BPB+EBPB.MEDIADESCRIPTOR]
000007D7 50                                 push    ax ; 02/09/2023
000007D8 24F0                               and     al, 0F0h
000007DA 3CF0                               cmp     al, 0F0h        ; allow for strange media
000007DC 58                                 pop     ax ; 02/09/2023
000007DD 751E                               jnz     short invalidbootsec

                                ; there were some (apparently a lot of them) diskettes that had been formatted
                                ; under dos 3.1 and earlier versions which have invalid bpbs in their boot
                                ; sectors. these are specifically diskettes that were formatted in drives
                                ; with one head, or whose side 0 was bad. these contain bpbs in the boot
                                ; sector that have the sec/clus field set to 2 instead of 1, as is standard
                                ; in dos. in order to support them, we have to introduce a "hack" that will
                                ; help our build bpb routine to recognise these specific cases, and to
                                ; set up out copy of the bpb accordingly.
                                ; we do this by checking to see if the boot sector is off a diskette that
                                ; is single-sided and is a pre-dos 3.20 diskette. if it is, we set the
                                ; sec/clus field to 1. if not, we carry on as normal.

                                checksinglesided:
                                            ;mov    al, [disksector+15h]
                                            ; 02/09/2023
                                            ; al = [disksector+15h]
000007DF 3CF0                               cmp     al, 0F0h
000007E1 741B                               jz      short gooddsk
000007E3 A801                               test    al, 1
000007E5 7517                               jnz     short gooddsk
000007E7 813E[5A01]332E                     cmp     word [disksector+8], 2E33h ; "3."
000007ED 7507                               jnz     short mustbeearlier
000007EF 803E[5C01]32                       cmp     byte [disksector+0Ah], 32h ; "2"
000007F4 7308                               jnb     short gooddsk

                                ; we must have a pre-3.20 diskette. set the sec/clus field to 1

                                mustbeearlier:
000007F6 C606[5F01]01                       mov     byte [disksector+0Dh], 1
                                                                    ; [disksector+EXT_BOOT.BPB+EBPB.SECTORSPERCLUSTER]
000007FB EB01                               jmp     short gooddsk
                                ; ----------------------------------------------------------------------------
```

```
                                invalidbootsec:
000007FD 43                             inc     bx              ; indicate that boot sector invalid
                                        ; 10/12/2022
                                movbpb_ret:
                                gooddsk:
000007FE F8                             clc
                                err_ret:
000007FF C3                             retn
                                ; ---------------------------------------------------------------------------

                                        ; 10/12/2022
                                ;err_ret:
                                        ;retn

                                ; =============== S U B   R O U T I N E =======================================

                                ; 15/10/2022
                                ; ---------------------------------------------------------------------------
                                ; 'movbpb' moves the bpb read from the boot sector into registers for use by
                                ; getbp routine at has1
                                ;
                                ; if the set_id_flag is 1, and if an extended boot record, then set volume
                                ; serial number, volume label, file system id in bds according to
                                ; the boot record. after that, this routine will set the set_id_flag to 2
                                ; to signal that volume label is set already from the extended boot record
                                ; (so, don't set it again by calling "set_volume_id" routine which uses
                                ; the volume label in the root directory.)
                                ; ---------------------------------------------------------------------------

                                ; 10/03/2019 - Retro DOS v4.0

                                ; 22/12/2023
                                %if 0
                                        ; 19/10/2022
                                movbpb:
                                        mov     dh, [disksector+0Dh]
                                                                ; disksector+EXT_BOOT.BPB+EBPB.SECTORSPERCLUSTER]
                                                                ; sectors per unit
                                        mov     bh, [disksector+11h]
                                                                ; [disksector+EXT_BOOT.BPB+EBPB.ROOTENTRIES]
                                                                ; number of directory entries
                                        mov     cx, [disksector+13h]
                                                                ; [disksector+EXT_BOOT.BPB+EBPB.TOTALSECTORS]
                                                                ; size of drive
                                        mov     ah, [disksector+15h]
                                                                ; [disksector+EXT_BOOT.BPB+EBPB.MEDIADESCRIPTOR]
                                                                ; media descriptor
                                        mov     al, [disksector+16h];
                                                                ; [disksector+EXT_BOOT.BPB+EBPB.SECTORSPERFAT]
                                                                ; number of fat       sectors
                                        mov     bl, [disksector+18h]
                                                                ; [disksector+EXT_BOOT.BPB+EBPB.SECTORSPERTRACK]
                                                                ; sectors per track
                                        mov     dl, [disksector+1Ah]
                                                                ; [disksector+EXT_BOOT.BPB+EBPB.HEADS]
                                                                ; number of heads
                                %else
                                        ; 29/12/2023
                                        ; 22/12/2023 - Retro DOS v5.0
                                        ; (PCDOS 7.1 IBMBIO.COM, BIOSCODE:0814h)
                                        ;;;
                                movbpb:
00000800 57                             push    di
00000801 83C706                         add     di, 6           ; BDS+6 = BDS.BPB
00000804 8D36[5D01]                     lea     si, [disksector+0Bh]
00000808 B93500                         mov     cx, 53          ; copy bios parameters block
                                                                ; from BPB_BytsPerSec to (FAT32) BS_DrvNum (excluded)
0000080B FC                             cld
0000080C F3A4                           rep movsb
0000080E 8B4CD3                         mov     cx, [si-45]     ; si = disksector+64 -> 64-45 = 19
                                                                ; disksektor+19 = BPB_TotSec16
00000811 31C0                           xor     ax, ax
00000813 E308                           jcxz    movbpb_bigdisk
00000815 26894DE0                       mov     [es:di-32], cx ; write 16 bit total sectors
                                                                ; to 32 bit total sectors field
00000819 268945E2                       mov     [es:di-30], ax ; BPB_TotalSec32+2 (BDS offset 29, BPB offset 23)
                                movbpb_bigdisk:
0000081D 3944D6                         cmp     [si-42], ax     ; BPB_FATSz16 = disksector+22
00000820 7410                           jz      short movbpb_fat32
                                movbpb_fat:
00000822 83EF1C                         sub     di, 28          ; di = BDS offset 31 (BPB offset 25)
                                        ; 29/12/2023
00000825 B90C00                         mov     cx, 12          ; clear 12 byte extended BDS (FAT32) fields
                                                                ; (which are used only for FAT32 disks)
00000828 F3AA                           rep stosb
0000082A 48                             dec     ax              ; -1 ; 0FFFFh
0000082B AB                             stosw                   ; set BDS offset 43 (dword) to -1
                                                                ; dword [BDS.BPB_FSInfo] = 0FFFFFFFFh
0000082C AB                             stosw
0000082D 40                             inc     ax              ; ax = 0
0000082E B10C                           mov     cl, 12
                                        ;mov    cx, 12          ; clear BDS offset 47 to 59
                                                                ; (BPB offset 41 to 53) (disksector offset 52 to 64)
00000830 F3AA                           rep stosb
                                movbpb_fat32:
00000832 5F                             pop     di
                                %endif
                                        ;;;

00000833 803E[9B04]01                   cmp     byte [set_id_flag], 1 ; called by get_bpb?
00000838 75C4                           jnz     short movbpb_ret
0000083A E81200                         call    mov_media_ids
0000083D 7205                           jb      short movbpb_conv ; conventional boot record?
0000083F C606[9B04]02                   mov     byte [set_id_flag], 2 ; signals that volume id is set
                                movbpb_conv:
00000844 803E[7700]01                   cmp     byte [fhave96], 1
00000849 75B3                           jnz     short movbpb_ret
0000084B E83714                         call    resetchanged    ; reset flags in bds to       not fchanged.
                                        ; 10/12/2022
                                        ; cf = 0
                                ;movbpb_ret:
                                        ;clc
0000084E C3                             retn

                                ; =============== S U B   R O U T I N E =======================================

                                ;copy the boot_serial number, volume id, and filesystem id from the
                                ;***extended boot record*** in ds:disksector to the bds table pointed
                                ;by es:di.

                                ;in.) es:di -> bds
                                ;     ds:disksector = valid extended boot record.
                                ;out.) vol_serial, bds_volid and bds_system_id in bds are set according to
                                ;      the boot record information.
```

```
14633                                 ;     carry flag set if not an extended bpb.
14634                                 ;     all registers saved except the flag.
14635                                 
14636                                 ; 22/12/2023
14637                                 %if 0
14638                                           ; 19/10/2022
14639                                 mov_media_ids:
14640                                           cmp     byte [disksector+26h], 29h
14641                                                                   ; [disksector+EXT_BOOT.SIG],
14642                                                                   ; EXT_BOOT_SIGNATURE
14643                                           jnz     short mmi_not_ext
14644                                           push    cx
14645                                           mov     cx, [disksector+27h]
14646                                                                   ; [disksector+EXT_BOOT.SERIAL]
14647                                           mov     [es:di+57h], cx       ; [es:di+BDS.vol_serial]
14648                                           mov     cx, [disksector+29h]
14649                                                                   ; [disksector+EXT_BOOT.SERIAL+2]
14650                                           mov     [es:di+59h], cx       ; [es:di+BDS.vol_serial+2]
14651                                           push    di
14652                                           push    si
14653                                           mov     cx, 11          ; size_of_EXT_BOOT_VOL_LABEL
14654                                           mov     si, disksector+2Bh
14655                                           ;mov    si, (offset disksector+2Bh) ;
14656                                                                   ; disksector+EXT_BOOT.VOL_LABEL
14657                                           add     di, 75          ; BDS.volid
14658                                           rep movsb
14659                                           ;mov    cx, 8           ; size_of_EXT_SYSTEM_ID
14660                                           ; 10/12/2022
14661                                           mov     cl, 8 ; cx = 8
14662                                           mov     si, disksector+36h
14663                                           ;mov    si, (offset disksector+36h) ; disksector+EXT_BOOT.SYSTEM_ID
14664                                           add     di, 5           ; (BDS.filesys_id-BDS.volid)-size_of_EXT_BOOT_VOL_LABEL
14665                                           rep movsb
14666                                           pop     si
14667                                           pop     di
14668                                           pop     cx
14669                                           ; 10/12/2022
14670                                           ; cf = 0
14671                                           ;clc           ; this clc is not required (16/06/2019 - Erdogan Tan)
14672                                                           ; (20/09/2022)
14673                                           retn
14674                                 %else
14675                                           ; 22/12/2023 - Retro DOS v5.0
14676                                           ; (PCDOS 7.1 IBMBIO.COM, BIOSCODE:0865h)
14677                                           ;;;
14678                                 mov_media_ids:
14679 0000084F 833E[6801]00                     cmp     word [disksector+16h], 0 ; BPB.FATSz16
14680 00000854 7507                             jnz     short mmi_chk_fat
14681 00000856 803E[9401]29                     cmp     byte [disksector+42h], 29h
14682                                                                   ; [disksector+FAT32_EXT_BOOT.SIG],
14683                                                                   ; EXT_BOOT_SIGNATURE
14684 0000085B EB05                             jmp     short mmi_chk_fat32
14685                                 mmi_chk_fat:
14686 0000085D 803E[7801]29                     cmp     byte [disksector+26h], 29h
14687                                                                   ; [disksector+EXT_BOOT.SIG],EXT_BOOT_SIGNATURE
14688                                 mmi_chk_fat32:
14689 00000862 7543                             jnz     short mmi_not_ext
14690 00000864 51                               push    cx
14691 00000865 50                               push    ax
14692 00000866 57                               push    di
14693 00000867 56                               push    si
14694 00000868 1E                               push    ds
14695 00000869 833E[6801]00                     cmp     word [disksector+16h], 0 ; BPB.FATSz16
14696 0000086E 750C                             jnz     short mmi_fat
14697
14698                                 mmi_fat32:                              ; FAT32 file system
14699                                           ;lds    cx, dword ptr ds:disksector+43h
14700 00000870 C50E[9501]                       lds     cx, [disksector+43h]  ; BS_FAT32_VolID
14701 00000874 BE[9901]                         mov     si, disksector+47h    ; BS_FAT32_VolLab
14702 00000877 B8[A401]                         mov     ax, disksector+52h    ; BS_FAT32_FilSysType
14703 0000087A EB0A                             jmp     short mmi_do
14704
14705                                 mmi_fat:
14706                                           ;lds    cx, dword ptr ds:disksector+27h
14707 0000087C C50E[7901]                       lds     cx, [disksector+27h]  ; BS_VolID
14708 00000880 BE[7D01]                         mov     si, disksector+2Bh    ; BS_VolLab
14709 00000883 B8[8801]                         mov     ax, disksector+36h    ; BS_FilSysType
14710                                 mmi_do:
14711 00000886 26898D8900                       mov     [es:di+89h], cx       ; [es:di+BDS.vol_serial]
14712                                                                   ; (BDS offset 137)
14713 0000088B 268C9D8B00                       mov     [es:di+8Bh], ds       ; [es:di+BDS.vol_serial+2]
14714 00000890 1F                               pop     ds
14715 00000891 B90B00                           mov     cx, 11
14716 00000894 83C77D                           add     di, 125         ; di = di+125 = BDS.volid
14717 00000897 F3A4                             rep movsb
14718 00000899 B108                             mov     cl, 8           ; di = di+136
14719 0000089B 89C6                             mov     si, ax          ; BS_FilSysType or BS_FAT32_FilSysType
14720 0000089D 83C705                           add     di, 5           ; di = di+141 = BDS.filesys_id
14721 000008A0 F3A4                             rep movsb
14722 000008A2 5E                               pop     si
14723 000008A3 5F                               pop     di
14724 000008A4 58                               pop     ax
14725 000008A5 59                               pop     cx
14726                                           ;clc    ; this clc is not required (16/06/2019 - Erdogan Tan)
14727                                                   ; (20/09/2022 - 27/06/2023) MSDOS 6.21 .. PCDOS 7.1
14728 000008A6 C3                               retn
14729                                 %endif
14730                                           ;;;
14731
14732                                 ; ---------------------------------------------------------------------------
14733
14734                                 mmi_not_ext:
14735 000008A7 F9                               stc
14736 000008A8 C3                               retn
14737
14738                                 ; =============== S U B   R O U T I N E =======================================
14739
14740                                 ; 15/10/2022
14741                                 ; ---------------------------------------------------------------
14742                                 ; read in the fat sector and get the media byte from it.
14743                                 ; input : es:di -> bds
14744                                 ; output:
14745                                 ;     carry set if an error occurs, ax contains error code.
14746                                 ;     otherwise, ah contains media byte on exit
14747                                 ; ---------------------------------------------------------------
14748
14749                                 readfat:
14750                                           ;mov    dh, 0
14751                                           ; 10/12/2022
14752 000008A9 30F6                             xor     dh, dh
14753 000008AB B90200                           mov     cx, 2           ; head 0
14754                                                                   ; cylinder 0, sector 2
14755 000008AE E80500                           call    read_sector
14756 000008B1 7202                             jb      short bad_fat_ret
```

```
000008B3 8A27                           mov     ah, [bx]        ; media byte
                                bad_fat_ret:
000008B5 C3                             retn

                                ; =============== S U B   R O U T I N E =======================================

                                ; 15/10/2022

                                ; ---------------------------------------------------------------------------
                                ; read a single sector into the temp buffer.
                                ; perform three retries in case of error.
                                ;   inputs: es:[di].bds_drivenum has physical drive to use
                                ;           cx has sector and cylinder
                                ;           dh has head
                                ;           es:di has bds
                                ;           ds has Bios_Data
                                ;
                                ;   outputs:        carry clear
                                ;                Bios_Data:bx point to sector
                                ;                   (note: some callers assume location of buffer)
                                ;
                                ;           carry set
                                ;               ax has rom error code
                                ;
                                ; register bp is preserved.
                                ; ---------------------------------------------------------------------------

                                ; 10/03/2019 - Retro DOS v4.0
                                ; 22/12/2023 - Retro DOS v5.0

                                        ; 19/10/2022
                                read_sector:
000008B6 55                             push    bp
000008B7 BD0300                         mov     bp, 3           ; make 3 attempts
000008BA 268A5504                       mov     dl, [es:di+4]   ; [es:di+BDS.drivenum]
000008BE BB[5201]                       mov     bx, disksector ; get es:bx to point to       buffer
                                rd_ret:
000008C1 06                             push    es
000008C2 1E                             push    ds
000008C3 07                             pop     es
000008C4 B80102                         mov     ax, 201h
000008C7 CD13                           int     13h             ; DISK - READ SECTORS INTO MEMORY
                                                                ; AL = number of sectors to read, CH = track, CL = sector
                                                                ; DH = head, DL       = drive, ES:BX -> buffer to fill
                                                                ; Return: CF set on error, AH =      status, AL = number of sectors read
000008C9 07                             pop     es
000008CA 734A                           jnb     short okret2
                                rd_rty:
000008CC E81205                         call    again           ; reset disk, decrement       bp, preserve ax
000008CF 7442                           jz      short err_rd_ret

                                        ; 22/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
000008D1 26F6453F01                     test    byte [es:di+3Fh], 1
                                        ;test   byte [es:di+23h], 1
                                        ;;test  byte ptr [es:di+23h], 1       ; [es:di+BDS.flags]
                                                                ; fnon_removable
000008D6 75E9                           jnz     short rd_ret
000008D8 803E[A905]00                   cmp     byte [media_set_for_format], 0
000008DD 7510                           jnz     short rd_skip1_dpt
000008DF 50                             push    ax
000008E0 1E                             push    ds              ; for retry, set the head settle time to 0Fh
000008E1 C536[2D01]                     lds     si, [dpt]
                                        ;mov    al, [si+9]      ; [si+DISK_PARMS.DISK_HEAD_STTL]
                                        ;mov    byte [si+9], 15 ; [si+DISK_PARMS.DISK_HEAD_STTL]
                                        ;                       ; NORMSETTLE
                                        ; 12/12/2022
000008E5 B00F                           mov     al, 15
000008E7 864409                         xchg    al, [si+9]
                                        ;
000008EA 1F                             pop     ds
000008EB A2[2A01]                       mov     [save_head_sttl], al
000008EE 58                             pop     ax
                                rd_skip1_dpt:
000008EF 06                             push    es
000008F0 1E                             push    ds
000008F1 07                             pop     es
000008F2 B80102                         mov     ax, 201h
000008F5 CD13                           int     13h             ; DISK - READ SECTORS INTO MEMORY
                                                                ; AL = number of sectors to read, CH = track, CL = sector
                                                                ; DH = head, DL       = drive, ES:BX -> buffer to fill
                                                                ; Return: CF set on error, AH =      status, AL = number of sectors read
000008F7 07                             pop     es
000008F8 9C                             pushf
000008F9 803E[A905]00                   cmp     byte [media_set_for_format], 0
000008FE 750E                           jnz     short rd_skip2_dpt
00000900 50                             push    ax
00000901 A0[2A01]                       mov     al, [save_head_sttl]
00000904 1E                             push    ds
00000905 C536[2D01]                     lds     si, [dpt]
00000909 884409                         mov     [si+9], al      ; [si+DISK_PARMS.DISK_HEAD_STTL]
0000090C 1F                             pop     ds
0000090D 58                             pop     ax
                                rd_skip2_dpt:
0000090E 9D                             popf
0000090F 7305                           jnb     short okret2
00000911 EBB9                           jmp     short rd_rty
                                ; ---------------------------------------------------------------------------

                                err_rd_ret:
00000913 B2FF                           mov     dl, 0FFh        ; make sure we ask rom if media      has changed
                                                                ; return error
00000915 F9                             stc

                                ; update information pertaining to last drive accessed, time of access, last
                                ; track accessed in that drive.

                                okret2:
00000916 8816[7600]                     mov     [step_drv], dl ; set up for head settle logic in disk
0000091A 8816[1E01]                     mov     [tim_drv], dl  ; save drive last accessed

                                        ; 22/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
0000091E 26886D78                       mov     [es:di+78h], ch
                                        ;mov    [es:di+46h], ch       ; [es:di+BDS.track]
                                                                ; save last track accessed on this drive
                                                                ; preserve flags in case error occurred
00000922 9C                             pushf
00000923 E89B04                         call    set_tim
00000926 9D                             popf                    ; restore flags
00000927 5D                             pop     bp
00000928 C3                             retn

                                ;---------------------------------------------------------------------------
                                ;   disk open/close routines
                                ;---------------------------------------------------------------------------
```

```
14881
14882                                   dsk_open:                       ; 2C7h:80Ah = 70h:2D7Ah
14883 00000929 803E[7700]00                       cmp     byte [fhave96], 0
14884 0000092E 7407                               jz      short dsk_open_exit ; done if no changeline support
14885 00000930 E871FC                             call    SetDrive        ; get bds for drive
14886                                             ; 22/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
14887 00000933 26FF453C                           inc     word [es:di+3Ch] ; [es:di+BDS.opcnt] ; BDS offset 60
14888                                             ;inc    word [es:di+20h] ; [es:di+BDS.opcnt]
14889                                   dsk_open_exit:
14890                                             ; 10/12/2022
14891                                             ; cf = 0
14892                                             ;clc            ; CF is already ZERO here (18/09/2022, MSDOS 5.0 IO.SYS)
14893                                                             ; (19/07/2019 -       Erdogan Tan - MSDOS 6.0       IO.SYS - retrodos4.s)
14894 00000937 C3                                 retn
14895                                   ; ---------------------------------------------------------------------------
14896
14897                                   dsk_close:                      ; 2C7h:81Ah = 70h:2D8Ah
14898 00000938 803E[7700]00                       cmp     byte [fhave96], 0
14899 0000093D 740E                               jz      short exitjx   ; done if no changeline      support
14900 0000093F E862FC                             call    SetDrive        ; get bds for drive
14901                                             ; 22/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
14902 00000942 26837D3C00                         cmp     word [es:di+3Ch], 0 ; [es:di+BDS.opcnt] ; BDS off 60
14903                                             ;cmp    word [es:di+20h], 0 ; [es:di+BDS.opcnt]
14904 00000947 7404                               jz      short exitjx   ; watch out for wrap
14905                                             ; 22/12/2023
14906 00000949 26FF4D3C                           dec     word [es:di+3Ch]
14907                                             ;dec    word [es:di+20h]
14908                                   exitjx:
14909                                             ; 10/12/2022
14910                                             ; cf = 0
14911                                             ;clc            ; CF is already ZERO here (18/09/2022, MSDOS 5.0 IO.SYS)
14912                                                             ; (19/07/2019 -       Erdogan Tan - MSDOS 6.0       IO.SYS - retrodos4.s)
14913 0000094D C3                                 retn
14914
14915                                   ;---------------------------------------------------------------------------
14916                                   ;       disk removable routine
14917                                   ;---------------------------------------------------------------------------
14918
14919                                             ; al is unit #
14920                                   dsk_rem:                        ; 2C7h:831h = 70h:2DA1h
14921 0000094E E853FC                             call    SetDrive        ; get bds for this drive
14922                                             ; 22/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
14923                                             ;test   byte [es:di+BDS.flags], fnon_removable
14924 00000951 26F6453F01                         test    byte [es:di+3Fh], 1 ; [es:di+BDS.flags], fnon_removable
14925 00000956 74F5                               jz      short exitjx
14926                                             ;test   byte [es:di+23h], 1 ; [es:di+BDS.flags], fnon_removable
14927                                             ;;jnz   short x_bus_exit ; non_rem
14928                                             ;jnz    short non_rem  ; 15/10/2022
14929                                             ;; 10/12/2022
14930                                             ;; cf = 0
14931                                             ;;clc                  ; CF is already ZERO here
14932                                             ;                      ; 15/10/2022
14933                                             ;retn
14934                                   ; ---------------------------------------------------------------------------
14935
14936                                   non_rem:
14937                                   x_bus_exit:
14938 00000958 B403                               mov     ah, 3          ; 2C7h:83Dh = 0070h:2DADh
14939                                                                    ; return busy status
14940 0000095A F9                                 stc
14941                                   dsk_ret:
14942 0000095B C3                                 retn
14943
14944                                   ;---------------------------------------------------------------------------
14945                                   ;       disk i/o routines
14946                                   ;---------------------------------------------------------------------------
14947
14948                                   dsk_writv:                      ; 2C7h:841h = 70h:2DB1h
14949                                             ;mov    word [wrtverify], 103h
14950                                             ; 19/10/2022
14951 0000095C C706[2001]0301                     mov     word [rflag], 103h
14952                                             ;mov    word ptr ds:rflag, 103h       ; write and verify
14953 00000962 EB06                               jmp     short dsk_cl
14954                                   ; ---------------------------------------------------------------------------
14955
14956                                   dsk_writ:                       ; 2C7h:849h = 70h:2DB9h
14957                                             ;mov    word [wrtverify], 3
14958                                             ; 19/10/2022
14959 00000964 C706[2001]0300                     mov     word [rflag], 3
14960                                             ;mov    word ptr ds:rflag, 3 ; romwrite
14961                                   dsk_cl:
14962 0000096A E8A400                             call    diskio         ; romwrite
14963                                   ; ---------------------------------------------------------------------------
14964
14965                                   dsk_io:
14966 0000096D 73EC                               jnb     short dsk_ret
14967 0000096F E965F7                             jmp     bc_err_cnt
14968                                   ; ---------------------------------------------------------------------------
14969
14970                                   dsk_read:                       ; ; 2C7h:857h =       70h:2DC7h
14971 00000972 E89700                             call    diskrd
14972 00000975 EBF6                               jmp     short dsk_io
14973
14974                                   ; =============== S U B   R O U TI N E =======================================
14975
14976                                   ; 15/10/2022
14977                                   ; 10/03/2019 - Retro DOS v4.0
14978                                   ; 22/12/2023 - Retro DOS v5.0
14979
14980                                   ;--------------------------------------------------------
14981                                   ; miscellaneous odd jump routines.
14982                                   ; moved out of mainline for speed.
14983
14984                                   ; if we have a system where we have virtual drives, we need
14985                                   ; to prompt the user to place the correct disk in the drive.
14986                                   ;
14987                                   ;   assume es:di -> bds, ds:->Bios_Data
14988                                   ;--------------------------------------------------------
14989
14990                                             ; 19/10/2022
14991                                   checksingle:
14992 00000977 50                                 push    ax
14993 00000978 53                                 push    bx
14994                                             ; 22/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
14995 00000979 268B5D3F                           mov     bx, [es:di+3Fh]       ; [es:di+BDS.flags]
14996                                             ;mov    bx, [es:di+23h]       ; [es:di+BDS.flags]
14997
14998                                   ; if hard drive, cannot change disk.
14999                                   ; if current owner of physical drive, no need to change diskette.
15000
15001 0000097D F6C321                             test    bl, 21h        ; fnon_removable|fi_own_physical
15002 00000980 7573                               jnz     short singleret
15003 00000982 F6C310                             test    bl, 10h        ; fi_am_mult
15004                                                                    ; is there a drive sharing this      physical drive?
```

```
15005 00000985 746E                             jz      short singleret
15006
15007                                   ; look for the previous owner of this physical drive
15008                                   ; and reset its ownership flag.
15009
15010 00000987 268A4504                         mov     al, [es:di+4]  ; [es:di+BDS.drivenum]
15011                                                                  ; get physical drive number
15012 0000098B 06                               push    es             ; preserve pointer to current bds
15013 0000098C 57                               push    di
15014 0000098D C43E[1901]                       les     di, [start_bds] ; get first bds
15015                                   scan_list:
15016 00000991 26384504                         cmp     [es:di+4], al
15017 00000995 7553                             jnz     short scan_skip       ; Not our drive. Try next bds.
15018 00000997 B320                             mov     bl, 20h ; ' '   ; fi_own_physical ; test ownership flag
15019                                           ; 22/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
15020 00000999 26845D3F                         test    [es:di+3Fh], bl       ; [es:di+BDS.flags]
15021                                           ;test   [es:di+23h], bl
15022 0000099D 744B                             jz      short scan_skip       ; he doesn't own it either. continue
15023 0000099F 26305D3F                         xor     [es:di+3Fh], bl
15024                                           ;xor    [es:di+23h], bl       ; reset ownership flag
15025 000009A3 5F                               pop     di             ; restore pointer to current bds
15026 000009A4 07                               pop     es
15027 000009A5 26085D3F                         or      [es:di+3Fh], bl
15028                                           ;or     [es:di+23h], bl       ; ; set ownership flag
15029
15030                                   ; we examine the fsetowner flag. if it is set, then we are using the code in
15031                                   ; checksingle to just set the owner of a drive. we must not issue the prompt
15032                                   ; in this case.
15033 000009A9 803E[7A00]01                     cmp     byte [fsetowner], 1
15034 000009AE 7517                             jnz     short not_fsetowner
15035                                           ;cmp    byte ptr es:[di+4], 0 ;      are we handling       drive number 0 ?
15036 000009B0 26807D0400                       cmp     byte [es:di+4], 0
15037 000009B5 753E                             jnz     short singleret
15038 000009B7 268A4505                         mov     al, [es:di+5]
15039                                           ;mov    al, es:[di+5]  ; [es:di+BDS.drivelet]
15040                                                                  ; get the DOS drive letter
15041 000009BB 06                               push    es
15042 000009BC 8E06[1A00]                       mov     es, [zeroseg]
15043 000009C0 26A20405                         mov     [es:LSTDRV], al
15044                                           ;mov    es:504h, al    ; [es:LSTDRV]
15045                                                                  ; set up sdsb
15046 000009C4 07                               pop     es             ; restore bds pointer
15047 000009C5 EB2E                             jmp     short singleret
15048                                   ; ---------------------------------------------------------------------------
15049
15050                                   ; to support "backward" compatibility with ibm's "single drive status byte"
15051                                   ; we now check to see if we are in a single drive system and the application
15052                                   ; has "cleverly" diddled the sdsb
15053
15054                                   not_fsetowner:
15055 000009C7 803E[7800]02                     cmp     byte [single], 2 ; if (single_drive_system)
15056 000009CC 7517                             jnz     short ignore_sdsb
15057 000009CE 50                               push    ax
15058 000009CF 268A4505                         mov     al, [es:di+5]  ; if (curr_drv == req_drv)
15059 000009D3 88C4                             mov     ah, al
15060 000009D5 06                               push    es
15061 000009D6 8E06[1A00]                       mov     es, [zeroseg]
15062 000009DA 2686060405                       xchg    al, [es:LSTDRV]
15063                                           ;xchg   al, es:504h    ; [es:LSTDRV]
15064                                                                  ; then swap(curr_drv,req_drv)
15065 000009DF 07                               pop     es
15066 000009E0 38C4                             cmp     ah, al         ; else
15067 000009E2 58                               pop     ax             ; swap(curr_drv,req_drv)
15068 000009E3 7410                             jz      short singleret       ; issue swap_dsk_msg
15069                                   ignore_sdsb:
15070 000009E5 E8B310                           call    swpdsk
15071 000009E8 EB0B                             jmp     short singleret
15072                                   ; ---------------------------------------------------------------------------
15073
15074                                   scan_skip:
15075 000009EA 26C43D                           les     di, [es:di]
15076                                           ;les    di, es:[di]    ; [es:di+BDS.link]
15077                                                                  ; go to next bds
15078 000009ED 83FFFF                           cmp     di, 0FFFFh ; -1       ; end of list?
15079 000009F0 759F                             jnz     short scan_list       ; continue until hit end of list
15080 000009F2 F9                               stc
15081 000009F3 5F                               pop     di             ; restore current bds
15082 000009F4 07                               pop     es
15083                                   singleret:
15084 000009F5 5B                               pop     bx
15085 000009F6 58                               pop     ax
15086 000009F7 C3                               retn
15087
15088                                   ; 22/12/2023
15089                                   %if 0
15090                                   ; ---------------------------------------------------------------------------
15091
15092                                   baddrive:
15093                                           mov     al, 8          ; sector not found
15094                                           jmp     short baddrive_ret
15095                                   %endif
15096
15097                                   ; ---------------------------------------------------------------------------
15098
15099                                   unformatteddrive:
15100 000009F8 B007                             mov     al, 7          ; unknown media
15101                                   ;baddrive_ret:
15102 000009FA F9                               stc
15103                                   ; ---------------------------------------------------------------------------
15104
15105                                   ioret:
15106 000009FB C3                               retn
15107
15108                                   ; ---------------------------------------------------------------------------
15109
15110                                           ; 22/12/2023 - Retro DOS v5.0
15111                                           ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0A1Bh
15112
15113 000009FC 10                       LBA_Packet: db 16               ; ...
15114                                                                  ; DAP buffer
15115 000009FD 00                                   db 0
15116 000009FE 0000                     dap_block_cnt:     dw 0                 ; ...
15117 00000A00 00000000                 dap_trans_buf:     dd 0                 ; ...
15118 00000A04 00000000                 dap_lba_value:     dd 0                 ; ...
15119 00000A08 00000000                         dd 0
15120
15121                                   ; ---------------------------------------------------------------------------
15122
15123                                   ; 15/10/2022
15124
15125                                   ; ---------------------------------------------------------------------------
15126                                   ;   disk i/o handler
15127                                   ;
15128                                   ;   al = drive number (0-6)
```

```
15129                                   ;   ah = media descriptor
15130                                   ;   cx = sector count
15131                                   ;   dx = first sector (low)
15132                                   ;   [start_sec_h] = first sector (high)  32 bit calculation.
15133                                   ;   ds = cs
15134                                   ;   es:di = transfer address
15135                                   ;   [rflag]=operation (2=read, 3=write)
15136                                   ;   [verify]=1 for verify after write
15137                                   ;
15138                                   ;   if successful carry flag = 0
15139                                   ;     else cf=1 and al contains error code
15140                                   ; ---------------------------------------------------------------------------
15141
15142                                           ; 12/12/2023
15143                                           ; ds = biosdata segment (cs = bioscode segment)
15144                                   diskrd:
15145                                           ;mov   ds:rflag, 2    ; romread
15146                                           ; 19/10/2022
15147 00000A0C C606[2001]02                     mov    byte [rflag], 2 ; romread
15148
15149                                   ; =============== S U B   R O U T I N E =======================================
15150
15151                                           ; 22/12/2023 - Retro DOS v5.0
15152                                           ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0A30h
15153                                   ; 22/12/2023
15154                                   %if 0
15155                                           ; 19/10/2022
15156                                   diskio:
15157                                           mov    bx, di         ; es:bx= transfer address
15158                                           mov    [xfer_seg], es ; save transfer       segment
15159                                           call   SetDrive
15160                                           mov    al, [es:di+10h]       ; [es:di+BDS.media]
15161                                           mov    [medbyt], al
15162                                           ;jcxz  short ioret
15163                                           jcxz   ioret
15164
15165                                   ; see if the media is formatted or not by checking the flags field in
15166                                   ; in the bds. if it is unformatted we cannot allow i/o, so we should
15167                                   ; go to the error exit at label unformatteddrive.
15168
15169                                           test   byte [es:di+24h], 2
15170                                           ;test  byte ptr es:[di+24h], 2      ; [es:di+BDS.flags+1]
15171                                                                  ; unformatted_media
15172                                           jnz    short unformatteddrive
15173                                           mov    [seccnt], cx   ; save sector count
15174                                           mov    [spsav], sp    ; save sp
15175
15176                                   ; ensure that we are trying to access valid sectors on the drive
15177
15178                                           mov    ax, dx
15179                                           xor    si, si ; 0
15180                                           add    dx, cx
15181                                           ;adc   si, 0
15182                                           ; 02/09/2023 (PCDOS 7.1)
15183                                           rcl    si, 1
15184                                           cmp    word [es:di+0Eh], 0 ; [es:di+BDS.totalsecs16]
15185                                                                  ; 32 bit sector ?
15186                                           jz     short sanity32
15187                                           ;cmp   si, 0
15188                                           ; 02/09/2023
15189                                           or     si, si
15190                                           jnz    short baddrive
15191                                           cmp    dx, [es:di+0Eh]       ; [es:di+BDS.totalsecs16]
15192                                           ja     short baddrive
15193                                           jmp    short sanityok
15194                                   ; ---------------------------------------------------------------------------
15195
15196                                   sanity32:
15197                                           add    si, [start_sec_h]
15198                                           cmp    si, [es:di+1Dh]       ; [es:di+BDS.totalsecs32+2]
15199                                           jb     short sanityok
15200                                           ja     short baddrive
15201                                           cmp    dx, [es:di+1Bh]       ; [es:di+BDS.totalsecs32]
15202                                           ja     short baddrive
15203                                   sanityok:
15204                                           mov    dx, [start_sec_h]
15205                                           add    ax, [es:di+17h]       ; [es:di+BDS.hiddensecs]
15206                                           adc    dx, [es:di+19h]       ; [es:di+BDS.hiddensecs+2]
15207
15208                                   ; now dx;ax have the physical first sector.
15209                                   ; since the following procedures is going to destroy ax, let's
15210                                   ; save it temporarily to saved_word.
15211
15212                                           mov    [saved_word], ax ; save the sector number (low)
15213
15214                                   ; set up pointer to disk base table in [dpt]. we cannot assume that iosetup
15215                                   ; will do it because we will skip the set up stuff with hard disks.
15216
15217                                           push   es
15218                                           ;mov   es, [zeroseg]
15219                                           ; 02/09/2023
15220                                           xor    si, si ; 0
15221                                           mov    es, si
15222                                           les    si, [es:DSKADR]
15223                                           ;les   si, es:78h     ; [es:DSKADR]
15224                                                                  ; current disk parm table
15225                                           mov    [dpt], si
15226                                           mov    [dpt+2], es
15227                                           pop    es
15228                                           test   byte [es:di+23h], 1 ; [es:di+BDS.flags]
15229                                                                  ; fnon_removable
15230                                           jnz    short skip_setup
15231                                           call   checksingle
15232
15233                                   ; check to see if we have previously noted a change line. the routine
15234                                   ; returns if everything is ok. otherwise, it pops off the stack and returns
15235                                   ; the proper error code.
15236
15237                                           cmp    byte [fhave96], 0 ; do we have changeline support?
15238                                           jz     short diskio_nochangeline ; brif not
15239                                           call   checklatchio   ; will do a sneaky pop stack return
15240                                                                  ; if a disk error occurs
15241                                   diskio_nochangeline:
15242                                           call   iosetup        ; set up tables       and variables for i/o
15243
15244                                   ; now the settle values are correct for the following code
15245
15246                                   skip_setup:
15247
15248                                   ; 32 bit sector calculation.
15249                                   ; dx:[saved_word] = starting sector number.
15250
15251                                           mov    ax, dx
15252                                           xor    dx, dx
```

```
                                ;div   word [es:di+13h] ; [es:di+BDS.secpertrack]
                                                        ; divide by sec per track
                                ; 02/09/2023
                                mov    cx, [es:di+13h]
                                div    cx
                                mov    [temp_h], ax
                                mov    ax, [saved_word]
                                div    cx ; 02/09/2023
                                ;div   word [es:di+13h] ; [es:di+BDS.secpertrack]
                                                        ; now, [temp_h]:ax = track #, dx = sector
                                ;inc   dl               ; sector number       is 1 based.
                                ; 18/12/2022
                                inc    dx
                                mov    [cursec], dl   ; save current sector
                                mov    cx, [es:di+15h]       ; es:di+BDS.heads]
                                                        ; get number of       heads
                                push   ax
                                xor    dx, dx
                                mov    ax, [temp_h]   ; divide tracks       by heads per cylinder
                                div    cx
                                mov    [temp_h], ax
                                pop    ax
                                div    cx              ; now, [temp_h]:ax = cylinder #, dx = head
                                cmp    word [temp_h], 0
                                ja     short baddrive_brdg
                                cmp    ax, 1024        ; 2^10 currently maxium      for track #.
                                ja     short baddrive_brdg
                                mov    [curhd], dl    ; save current head
                                mov    [curtrk], ax   ; save current track

                        ; we are now set up for the i/o. normally, we consider the dma boundary
                        ; violations here. not true. we perform the operation as if everything is
                        ; symmetric; let the int 13 handler worry about the dma violations.

                                mov    ax, [seccnt]
                                call   block          ; (cas - call/ret)
                                ;call  done
                                ;retn
                                ; 18/12/2022
                                jmp    done
                        %else
                                ;;;    ; 22/12/2023
                        diskio:
00000A11 89FB                   mov    bx, di         ; al = drive number
                                                      ; cx = sector count
                                                      ; dx = first sector (low)
                                                      ; [start_sec_h] = first sector (high)
                                                      ;
                                                      ; es:bx = transfer address
00000A13 8C06[A804]             mov    [xfer_seg], es ; save transfer segment
00000A17 E88AFB                 call   SetDrive
00000A1A 268A4510               mov    al, [es:di+10h]       ; [es:di+BDS.media]
00000A1E A2[1F01]               mov    [medbyt], al
00000A21 E3D8                   jcxz   ioret

                        ; see if the media is formatted or not by checking the flags field in
                        ; in the bds. if it is unformatted we cannot allow i/o, so we should
                        ; go to the error exit at label unformatteddrive.

00000A23 26F6454002             test   byte [es:di+40h], 2 ; [es:di+BDS.flags+1]
                                                      ; unformatted_media
00000A28 75CE                   jnz    short unformatteddrive
00000A2A 890E[2201]             mov    [seccnt], cx   ; save sector count
00000A2E 8926[3501]             mov    [spsav], sp    ; save sp

                        ; ensure that we are trying to access valid sectors on the drive

00000A32 89D0                   mov    ax, dx
00000A34 31F6                   xor    si, si ; 0
00000A36 01CA                   add    dx, cx
00000A38 D1D6                   rcl    si, 1
00000A3A 26837D0E00             cmp    word [es:di+0Eh], 0 ; [es:di+BDS.totalsecs16]
                                                      ; > 32 bit sector ?
00000A3F 740E                   jz     short sanity32
00000A41 09F6                   or     si, si
00000A43 7506                   jnz    short baddrive
00000A45 263B550E               cmp    dx, [es:di+0Eh]       ; [es:di+BDS.totalsecs16]
                                ;ja    short baddrive
                                ;jmp   short sanityok
                                ; 22/12/2023
00000A49 7616                   jna    short sanityok
                        ; 29/12/2023
                        ; 22/12/2023
                        ;%if 1
                        ; ---------------------------------------------------------------------------

                        baddrive:
00000A4B B008                   mov    al, 8          ; sector not found
                                ;jmp   short baddrive_ret
                        ; ---------------------------------------------------------------------------
                        ;unformatteddrive:
                                ;mov   al, 7          ; unknown media
                        baddrive_ret:
00000A4D F9                     stc
                        ;ioret:
00000A4E C3                     retn
                        ;%endif

                        ; ---------------------------------------------------------------------------

                        sanity32:
00000A4F 0336[9C04]             add    si, [start_sec_h]
00000A53 263B751D               cmp    si, [es:di+1Dh]       ; [es:di+BDS.totalsecs32+2]
00000A57 7208                   jb     short sanityok
00000A59 77F0                   ja     short baddrive
00000A5B 263B551B               cmp    dx, [es:di+1Bh]       ; [es:di+BDS.totalsecs32]
00000A5F 77EA                   ja     short baddrive
                        sanityok:
00000A61 8B16[9C04]             mov    dx, [start_sec_h]
00000A65 26034517               add    ax, [es:di+17h]       ; [es:di+BDS.hiddensecs]
00000A69 26135519               adc    dx, [es:di+19h]       ; [es:di+BDS.hiddensecs+2]

                        ; now dx;ax have the physical first sector.
                        ; since the following procedures is going to destroy ax, let's
                        ; save it temporarily to saved_word.

00000A6D A3[9E04]               mov    [saved_word], ax ; save the sector number (low)

                        ; set up pointer to disk base table in [dpt]. we cannot assume that iosetup
                        ; will do it because we will skip the set up stuff with hard disks.

00000A70 06                     push   es
00000A71 31F6                   xor    si, si ; 0
00000A73 8EC6                   mov    es, si
```

```
15377                                   ;les    si, dword ptr es:78h
15378 00000A75 26C4367800               les     si, [es:78h]    ; INT 1Eh vector address
15379                                                           ; [es:DSKADR] - current disk parm table
15380 00000A7A 8936[2D01]               mov     [dpt], si
15381 00000A7E 8C06[2F01]               mov     [dpt+2], es
15382 00000A82 07                       pop     es
15383 00000A83 26F6453F01               test    byte [es:di+3Fh], 1 ; [es:di+BDS.flags], fnon_removable
15384 00000A88 7510                     jnz     short chk_13h_ext_flag
15385 00000A8A E8EAFE                   call    checksingle
15386
15387                                 ; check to see if we have previously noted a change line. the routine
15388                                 ; returns if everything is ok. otherwise, it pops off the stack and returns
15389                                 ; the proper error code.
15390
15391 00000A8D 803E[7700]00                      cmp     byte [fhave96], 0 ; do we have changeline support?
15392 00000A92 7403                              jz      short diskio_nochangeline ; brif not
15393 00000A94 E8D210                           call    checklatchio    ; will do a sneaky pop stack return
15394                                                                   ; if a disk error occurs
15395                                 diskio_nochangeline:
15396 00000A97 E8E000                           call    iosetup         ; set up tables and variables for i/o
15397
15398                                 chk_13h_ext_flag:
15399 00000A9A 26F6454004                       test    byte [es:di+40h], 4 ; [es:di+BDS.flags+1], fLBArw
15400                                                                   ; LBA read/write flag
15401 00000A9F 7539                             jnz     short set_lbarw_1
15402                                           ;jmp    skip_setup
15403                                           ; 22/12/2023
15404                                 ; ---------------------------------------------------------------------------
15405
15406                                 ; now the settle values are correct for the following code
15407
15408                                 skip_setup:
15409
15410                                 ; 32 bit sector calculation.
15411                                 ; dx:[saved_word] = starting sector number.
15412
15413                                           ;push   bp ; ! (not necessary) ; 22/12/2023
15414 00000AA1 92                               xchg    ax, dx ; mov ax,dx
15415 00000AA2 31D2                             xor     dx, dx
15416 00000AA4 268B4D13                         mov     cx, [es:di+13h]         ; [es:di+BDS.secpertrack]
15417                                                                   ; divide by sec per track
15418 00000AA8 F7F1                             div     cx
15419 00000AAA 95                               xchg    ax, bp ; mov bp,ax
15420 00000AAB A1[9E04]                         mov     ax, [saved_word]
15421 00000AAE F7F1                             div     cx              ; [es:di+BDS.secpertrack]
15422                                                                   ; now, bp:ax = track #, dx = sector
15423                                                                   ; sector number is 1 based.
15424 00000AB0 42                               inc     dx
15425 00000AB1 8816[3101]                       mov     [cursec], dl    ; save current sector
15426 00000AB5 268B4D15                         mov     cx, [es:di+15h] ; [es:di+BDS.heads]
15427                                                                   ; get number of heads
15428                                           ; 22/12/2023
15429                                           ;push   ax ; *
15430 00000AB9 31D2                             xor     dx, dx
15431 00000ABB 95                               xchg    ax, bp ; bp = *         ; divide tracks by heads per cylinder
15432 00000ABC F7F1                             div     cx
15433 00000ABE 95                               xchg    ax, bp ; ax = *, bp = **
15434                                           ;pop    ax ; *
15435 00000ABF F7F1                             div     cx              ; now, bp:ax = cylinder #, dx = head
15436 00000AC1 09ED                             or      bp, bp ; ** = 0 ?
15437                                           ;pop    bp ; ! ; 22/12/2023
15438                                           ;jnz    short baddrive_brdg
15439                                           ; 22/12/2023
15440 00000AC3 7586                             jnz     short baddrive
15441
15442                                           ;cmp    ax, 1024        ; 2^10 currently maximum for track #.
15443                                           ;jnb    short baddrive_brdg
15444                                           ; 22/12/2023
15445 00000AC5 80FC04                           cmp     ah, 4   ; if ax >= 4*256 (1024)
15446 00000AC8 7381                             jnb     short baddrive
15447
15448 00000ACA 8816[3201]                       mov     [curhd], dl     ; save current head
15449 00000ACE A3[3301]                         mov     [curtrk], ax    ; save current track
15450
15451                                 ; we are now set up for the i/o. normally, we consider the dma boundary
15452                                 ; violations here. not true. we perform the operation as if everything is
15453                                 ; symmetric; let the int 13 handler worry about the dma violations.
15454
15455 00000AD1 A1[2201]                           mov     ax, [seccnt]
15456 00000AD4 E81F01                           call    block
15457                                           ;call   done
15458                                           ;retn
15459                                           ; 22/12/2023
15460 00000AD7 E9E500                           jmp     done
15461
15462                                 ; ---------------------------------------------------------------------------
15463
15464                                 set_lbarw_1:
15465 00000ADA A1[9E04]                         mov     ax, [saved_word] ; check for mini disk
15466                                                                   ; (logical dos drive/partition)
15467 00000ADD 26837D7901                       cmp     word [es:di+79h], 1 ; [di+BDS.bdsm_ismini]
15468                                                                    ; logical dos partition
15469 00000AE2 750F                             jnz     short set_lbarw_2 ; not a logical dos partition/drive
15470 00000AE4 26837D7B00                       cmp     word [es:di+7Bh], 0 ; [di+BDS.bdsm_hidden_trks] (> 0)
15471 00000AE9 7408                             jz      short set_lbarw_2
15472 00000AEB 26034517                         add     ax, [es:di+17h]         ; [es:di+BDS.hiddensecs]
15473 00000AEF 26135519                         adc     dx, [es:di+19h]         ; [es:di+BDS.hiddensecs+2]
15474
15475                                 set_lbarw_2:
15476 00000AF3 2EA3[040A]                       mov     [cs:dap_lba_value], ax
15477 00000AF7 2E8916[060A]                     mov     [cs:dap_lba_value+2], dx
15478 00000AFC 2E891E[000A]                     mov     [cs:dap_trans_buf], bx
15479 00000B01 A1[A804]                         mov     ax, [xfer_seg]
15480 00000B04 2EA3[020A]                       mov     [cs:dap_trans_buf+2], ax
15481 00000B08 A1[2201]                         mov     ax, [seccnt]
15482 00000B0B 2EA3[FE09]                       mov     [cs:dap_block_cnt], ax
15483 00000B0F BD0500                           mov     bp, 5
15484 00000B12 892E[A304]                       mov     [vretry_cnt], bp ; verify op. retry cnt for write-verify
15485 00000B16 892E[A504]                       mov     [soft_ecc_cnt], bp ; soft ecc error retry count
15486
15487                                 set_lbarw_3:
15488 00000B1A 268A5504                         mov     dl, [es:di+4]  ; [es:di+BDS.drivenum]
15489 00000B1E 8A26[2001]                       mov     ah, [rflag]    ; get read/write indicator
15490 00000B22 80C440                           add     ah, 40h
15491 00000B25 30C0                             xor     al, al
15492 00000B27 1E                               push    ds
15493 00000B28 0E                               push    cs
15494 00000B29 1F                               pop     ds
15495 00000B2A BE[FC09]                         mov     si, LBA_Packet
15496 00000B2D CD13                             int     13h             ; LBA read/write
15497 00000B2F 1F                               pop     ds
15498 00000B30 731A                             jnc     short set_lbarw_7
15499 00000B32 E8AC02                           call    again
15500                                 set_lbarw_9:
```

```
15501 00000B35 7503                     jnz     short set_lbarw_4
15502 00000B37 E92B02                   jmp     harderr
15503                           ; ---------------------------------------------------------------------------
15504
15505                           set_lbarw_4:
15506                           ;set_lbarw_9:      ; 22/12/2023
15507 00000B3A 80FCCC                   cmp     ah, 0CCh        ; Write fault (hard disk)
15508 00000B3D 7505                     jnz     short set_lbarw_5
15509 00000B3F BD0100                   mov     bp, 1
15510                                   ;jmp    short set_lbarw_6
15511                                   ; 17/04/2024
15512 00000B42 EBD6                     jmp     short set_lbarw_3
15513                           ; ---------------------------------------------------------------------------
15514
15515                           set_lbarw_5:
15516                           set_lbarw_10:      ; 22/12/2023
15517 00000B44 C706[A504]0500           mov     word [soft_ecc_cnt], 5 ; soft ecc error retry count
15518                           set_lbarw_6:
15519                           set_lbarw_11:
15520 00000B4A EBCE                     jmp     short set_lbarw_3
15521                           ; ---------------------------------------------------------------------------
15522
15523                           set_lbarw_7:
15524 00000B4C 813E[2001]0301           cmp     word [rflag], 103h
15525 00000B52 7523                     jnz     short set_lbarw_12
15526 00000B54 B444                     mov     ah, 44h
15527 00000B56 1E                       push    ds
15528 00000B57 0E                       push    cs
15529 00000B58 1F                       pop     ds
15530 00000B59 CD13                     int     13h             ; DISK - IBM/MS Extension - VERIFY SECTORS
15531                                                           ;  (DL - drive, [SI - disk address packet)
15532 00000B5B 1F                       pop     ds
15533 00000B5C 7319                     jnc     short set_lbarw_12
15534 00000B5E 80FC11                   cmp     ah, 11h         ; ECC corrected data error (soft error - retried OK )
15535 00000B61 7506                     jnz     short set_lbarw_8
15536 00000B63 FF0E[A504]               dec     word [soft_ecc_cnt]
15537                           ;set_lbarw_8:
15538 00000B67 740E                     jz      short set_lbarw_12
15539                           set_lbarw_8:
15540 00000B69 E8CF07                   call    ResetDisk
15541 00000B6C 80FC11                   cmp     ah, 11h
15542 00000B6F 74D9                     jz      short set_lbarw_11
15543 00000B71 FF0E[A304]               dec     word [vretry_cnt]
15544                                   ;jnz    short set_lbarw_9
15545                                   ;jmp    harderr
15546                                   ; 22/12/2023
15547 00000B75 EBBE                     jmp     short set_lbarw_9
15548
15549                           ; ---------------------------------------------------------------------------
15550                           ;               ; 22/12/2023
15551                           ;set_lbarw_9:
15552                           ;               cmp     ah, 0CCh
15553                           ;               jnz     short set_lbarw_10
15554                           ;               mov     bp, 1
15555                           ;               jmp     short set_lbarw_11
15556                           ; ---------------------------------------------------------------------------
15557                           ;               ; 22/12/2023
15558                           ;set_lbarw_10:
15559                           ;               mov     word [soft_ecc_cnt], 5 ; soft ecc error retry count
15560                           ;set_lbarw_11:
15561                           ;               jmp     short set_lbarw_3
15562                           ; ---------------------------------------------------------------------------
15563
15564                           set_lbarw_12:
15565 00000B77 31C0                     xor     ax, ax
15566                           skip_dpt_setting:  ; 23/12/2023
15567 00000B79 C3                       retn
15568                                   ;;;     ; 22/12/2023
15569                           %endif
15570
15571                           ; ---------------------------------------------------------------------------
15572
15573                                   ; 22/12/2023
15574                           ;baddrive_brdg:
15575                                   ;jmp    baddrive
15576
15577                           ; =============== S U B   R O U T I N E =======================================
15578
15579                           ;--------------------------------------------------------------
15580                           ; set the drive-last-accessed flag for diskette only.
15581                           ; we know that the hard disk will not be removed.
15582                           ; es:di -> current bds.
15583                           ; ds -> Bios_Data
15584                           ; ax,cx,si are destroyed.
15585                           ;--------------------------------------------------------------
15586
15587                                   ; 23/12/2023 - Retro DOS v5.0
15588
15589                                   ; 19/10/2022
15590                           iosetup:
15591 00000B7A 268A4504                 mov     al, [es:di+4]  ; [es:di+BDS.drivenum]
15592 00000B7E A2[1E01]                 mov     [tim_drv], al  ; save drive letter
15593
15594                           ; determine proper head settle values
15595
15596 00000B81 803E[A905]00             cmp     byte [media_set_for_format], 0
15597 00000B86 75F1                     jnz     short skip_dpt_setting
15598 00000B88 A0[2C01]                 mov     al, [eot]       ; fetch up eot before changing ds
15599 00000B8B 1E                       push    ds
15600 00000B8C C536[2D01]               lds     si, [dpt]       ; get pointer to disk base table
15601 00000B90 884404                   mov     [si+4], al
15602
15603                                   ;; 23/12/2023
15604                                   ;mov    ah, al
15605                                   ;mov    al, [si+10]     ; [si+DISK_PARMS.DISK_MOTOR_STRT]
15606                                   ;;mov   ah, [si+4]      ; [si+DISK_PARMS.DISK_EOT]
15607                                   ;pop    ds
15608                                   ;mov    [motorstartup], al
15609                                   ;mov    [save_eot], ah
15610                                   ; 06/04/2024
15611 00000B93 8A640A                   mov     ah,[si+10]
15612 00000B96 1F                       pop     ds
15613 00000B97 8826[2601]               mov     [motorstartup], ah
15614 00000B9B A2[2B01]                 mov     [save_eot], al
15615
15616                           ; for 3.5" drives, both external as well as on the k09, we need to set the
15617                           ; motor start time to 4. this checking for every i/o is going to affect
15618                           ; performance across the board, but is necessary!!
15619
15620 00000B9E 1E                       push    ds
15621 00000B9F C536[2D01]               lds     si, [dpt]       ; get pointer to disk base table
15622                                   ; 23/12/2023  - Retro DOS v5.0
15623 00000BA3 26807D3E02               cmp     byte [es:di+3Eh], 2 ; (PCDOS 7.1 IBMBIO.COM)
15624                                   ;cmp    byte [es:di+22h], 2 ; [es:di+BDS.formfactor]
```

```
                                                        ; ffSmall
00000BA8 7505                           jnz     short motor_start_ok
00000BAA B004                           mov     al, 4
00000BAC 86440A                         xchg    al, [si+10]     ; [si+DISK_PARMS.DISK_MOTOR_STRT]
                                motor_start_ok:

                                ; ds:si now points to disk parameter table.
                                ; get current settle and set fast settle

                                        ;xor     al, al
                                        ;inc     al             ; ibm wants fast settle      to be 1
                                        ; 18/12/2022
00000BAF 31C0                           xor     ax, ax
00000BB1 40                             inc     ax
00000BB2 864409                         xchg    al, [si+9]      ; [si+DISK_PARMS.DISK_HEAD_STTL]
                                                                ; get settle and set up      for fast
00000BB5 1F                             pop     ds
00000BB6 A2[2701]                       mov     [settlecurrent], al
00000BB9 B00F                           mov     al, 15          ; NORMSETTLE
                                                                ; someone has diddled the settle
00000BBB A2[2801]                       mov     [settleslow], al
                                        ; 23/12/2023
                                ;skip_dpt_setting:
00000BBE C3                             retn

                                ; =============== S U B   R O U TI N E =======================================

                                ;-----------------------------------------------------------
                                ; set time of last access, and reset default values in the dpt.
                                ;
                                ;     note: trashes (at least) si
                                ;-----------------------------------------------------------

                                        ; 23/12/2023 - Retro DOS v5.0

                                        ; 19/10/2022
                                done:
                                        ;test    byte [es:di+23h], 1 ; [es:di+BDS.flags]
                                                                     ; fnon_removable
                                        ; 23/12/2023
00000BBF 26F6453F01                     test    byte [es:di+3Fh], 1 ; (PCDOS 7.1 IBMBIO.COM)
00000BC4 752F                           jnz     short ddbx      ; do not set for non-removable media
00000BC6 E8F801                         call    set_tim
                                ;diddleback:
                                ; 09/12/2022
                                diddle_back:
00000BC9 9C                             pushf
00000BCA 803E[A905]00                   cmp     byte [media_set_for_format], 0
00000BCF 7523                           jnz     short nodiddleback
00000BD1 50                             push    ax
00000BD2 06                             push    es
00000BD3 C436[2D01]                     les     si, [dpt]
00000BD7 A0[2B01]                       mov     al, [save_eot]
00000BDA 26884404                       mov     [es:si+4], al  ; [es:si+DISK_PARMS.DISK_EOT]
00000BDE A0[2701]                       mov     al, [settlecurrent]
00000BE1 8A26[2601]                     mov     ah, [motorstartup]
00000BE5 26884409                       mov     [es:si+9], al  ; [es:si+DISK_PARMS.DISK_HEAD_STTL]
00000BE9 26C6440302                     mov     byte [es:si+3], 2 ; [es:si+DISK_PARMS.DISK_SECTOR_SIZ]
00000BEE 2688640A                       mov     [es:si+0Ah], ah       ; [es:si+DISK_PARMS.DISK_MOTOR_STRT]
00000BF2 07                             pop     es
00000BF3 58                             pop     ax
                                nodiddleback:
00000BF4 9D                             popf
                                ddbx:
00000BF5 C3                             retn

                                ; =============== S U B   R O U TI N E =======================================

                                ;-----------------------------------------------------------
                                ;read the number of sectors specified in ax,
                                ;handling track boundaries
                                ;es:di -> bds for this drive
                                ;-----------------------------------------------------------

                                        ; 23/12/2023 - Retro DOS v5.0

                                        ; 19/10/2022
                                block:
00000BF6 09C0                           or      ax, ax
00000BF8 74FB                           jz      short ddbx
                                        ; 23/12/2023
00000BFA 26F6453F01                     test    byte [es:di+3Fh], 1 ; (PCDOS 7.1 IBMBIO.COM)
                                        ;test    byte [es:di+23h], 1 ; [es:di+BDS.flags]
                                                                     ; fnon_removable
00000BFF 740D                           jz      short block_floppy

                                ; check     to see if multi       track operation       is allowed. if not
                                ; we have to go     to the block_floppy below to break up the operation.

00000C01 F606[A004]80                   test    byte [multrk_flag], 80h
                                        ;test    byte ptr ds:multrk_flag, 80h ; multrk_on
00000C06 7406                           jz      short block_floppy
00000C08 E82800                         call    Disk
00000C0B 31C0                           xor     ax, ax
00000C0D C3                             retn
                                ; ---------------------------------------------------------------------------

                                block_floppy:

                                ; read at most 1 track worth. perform minimization at sector / track

00000C0E 268A4D13                       mov     cl, [es:di+19] ; [es:di+BDS.secpertrack]
                                        ;inc     cl
                                        ; 23/12/2023
00000C12 41                             inc     cx
00000C13 2A0E[3101]                     sub     cl, [cursec]
00000C17 30ED                           xor     ch, ch
00000C19 39C8                           cmp     ax, cx
00000C1B 7302                           jnb     short gotmin
00000C1D 89C1                           mov     cx, ax
                                gotmin:

                                ; ax is the requested number of sectors to read
                                ; cx is the number that we can do on this track

00000C1F 50                             push    ax
00000C20 51                             push    cx
00000C21 89C8                           mov     ax, cx
00000C23 E80D00                         call    Disk
00000C26 59                             pop     cx
00000C27 58                             pop     ax

                                ; cx is the number of sectors just transferred

```

```
00000C28 29C8                           sub     ax, cx          ; reduce sectors-remaining by last i/o
00000C2A D0E1                           shl     cl, 1
00000C2C 00CF                           add     bh, cl          ; adjust transfer address
00000C2E EBC6                           jmp     short block
                                dskerr_brdg:
00000C30 E9F400                         jmp     dskerr

                                ; =============== S U B   R O U T I N E =======================================

                                ; 15/10/2022

                                ;--------------------------------------------------------------
                                ;perform disk i/o with retries
                                ; al = number of sectors (1-8, all on one track)
                                ; es:di point to drive parameters
                                ; xfer_seg:bx = transfer address
                                ;               (must not cross a 64k physical boundary)
                                ; [rflag] = 2 if read, 3 if write
                                ; [verify] = 0 for normal, 1 for verify after write
                                ;--------------------------------------------------------------

                                        ; 18/04/2024
                                        ; 23/12/2023 - Retro DOS v5.0
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0C74h)

                                        ; 19/10/2022
                                Disk:

                                ; Check for hard disk format and
                                ; if TRUE then set max error count to 2

00000C33 BD0500                         mov     bp, 5           ; MAXERR
                                                                ; set up retry count
                                        ; 18/04/2024
                                        ;; 23/12/2023
                                        ;mov    cl, [es:di+3Fh]
                                        ;and    cx, 1
00000C36 26F6453F01                     test    byte [es:di+3Fh], 1
                                        ;test   byte [es:di+23h], 1
                                                                ; [es:di+BDS.flags], fnon_removable
00000C3B 7408                           jz      short GetRdWrInd
00000C3D 80FC04                         cmp     ah, 4           ; romverify ; Is this a       track verify?
00000C40 7403                           jz      short GetRdWrInd
00000C42 BD0200                         mov     bp, 2           ; This is not verify so only 1 retry
                                GetRdWrInd:
00000C45 892E[A304]                     mov     [vretry_cnt], bp ; verify op. retry cnt for write-verify
00000C49 892E[A504]                     mov     [soft_ecc_cnt], bp ; soft ecc error retry count.
00000C4D 8A26[2001]                     mov     ah, [rflag]     ; get read/write indicator
                                ;retry:
                                ; 09/12/2022
                                _retry:
00000C51 50                             push    ax
00000C52 8B16[3301]                     mov     dx, [curtrk]
                                        ;; 23/12/2023
                                        ;jcxz   disk_not_mini
                                        ; 18/04/2024
00000C56 26F6453F01                     test    byte [es:di+3Fh], 1
                                        ;test   byte [es:di+23h], 1
00000C5B 740B                           jz      short disk_not_mini

                                        ; 23/12/2023
00000C5D 26837D7901                     cmp     word [es:di+79h], 1
                                        ;cmp    word [es:di+47h], 1 ; [es:di+BDS.bdsm_ismini]
                                                                ; is this a mini disk? ((logical dos partition))
00000C62 7504                           jnz     short disk_not_mini ; no. continue to next.
                                        ; 23/12/2023
00000C64 2603557B                       add     dx, [es:di+7Bh]
                                        ;add    dx, [es:di+49h]       ; [es:di+BDS.bdsm_hidden_trks]
                                                                ; add hidden trks.
                                disk_not_mini:
00000C68 D0CE                           ror     dh, 1
00000C6A D0CE                           ror     dh, 1
00000C6C 0A36[3101]                     or      dh, [cursec]
00000C70 89D1                           mov     cx, dx
00000C72 86CD                           xchg    ch, cl          ;  cl = sector, ch = cylinder
00000C74 8A36[3201]                     mov     dh, [curhd]     ; load current head number and
00000C78 268A5504                       mov     dl, [es:di+4]   ; physical drive number
                                                                ; [es:di+BDS.drivenum]
                                        ; 23/12/2023
00000C7C 26807D3E05                     cmp     byte [es:di+3Eh], 5
                                        ;cmp    byte [es:di+22h], 5 ; [es:di+BDS.formfactor], ffHardFile
00000C81 7411                           jz      short do_fast  ; hard files use fast speed

                                ; if we have [step_drv] set to -1, we use the slow settle time.
                                ; this helps when we have just done a reset disk operation and the head has
                                ; been moved to another cylinder - the problem crops up with 3.5" drives.

00000C83 803E[7600]FF                   cmp     byte [step_drv], 0FFh ; -1
                                        ;jz     short do_writej
                                        ; 23/12/2023
00000C88 746A                           jz      short do_write
00000C8A 80FC02                         cmp     ah, 2           ; romread
00000C8D 7405                           jz      short do_fast
00000C8F 80FC04                         cmp     ah, 4           ; romverify
                                        ;jz     short do_fast
                                        ; 23/12/2023
00000C92 7560                           jnz     short do_write
                                ;do_writej:

                                ; reads always fast, unless we have just done a disk reset operation

                                        ;jmp    short do_write ; reads always fast
                                ; ---------------------------------------------------------------------------

                                do_fast:
00000C94 E80501                         call    fastspeed       ; change settle       mode
                                testerr:
00000C97 7297                           jb      short dskerr_brdg

                                        ; 23/12/2023 Retro DOS v5.0
                                        ; (PCDOS 7.1 IBMBIO.COM)
00000C99 83FD05                         cmp     bp, 5           ; is there retry ?
00000C9C 7505                           jnz     short testerror         ; yes
00000C9E 80FCBB                         cmp     ah, 0BBh        ; Undefined error (hard disk)
00000CA1 748D                           jz      short dskerr_brdg
                                testerror:

                                ; set drive and track of last access

00000CA3 8816[7600]                     mov     [step_drv], dl
                                        ; 23/12/2023
00000CA7 26886D78                       mov     [es:di+78h], ch
                                        ;mov    [es:di+46h], ch       ; [es:di+BDS.track]
                                no_set:
```

```
15873                                           ;cmp    word [wrtverify], 103h
15874 00000CAB 813E[2001]0301                   cmp     word [rflag], 103h ; check for write and verify
15875 00000CB1 7452                             jz      short doverify
15876                                   noverify:
15877 00000CB3 58                               pop     ax
15878
15879                                   ; check the flags word in the bds to see if the drive is non removable
15880                                   ; if not we needn't do anything special
15881                                   ; if it is a hard disk then check to see if multi-track operation
15882                                   ; is specified. if specified we don't have to calculate for the next
15883                                   ; track since we are already done. so we can go to the exit of this routine.
15884
15885                                           ; 23/12/2023
15886 00000CB4 26F6453F01                       test    byte [es:di+3Fh], 1
15887                                           ;test   byte [es:di+23h], 1 ; [es:di+BDS.flags]
15888                                                                   ; fnon_removable
15889 00000CB9 7407                             jz      short its_removable
15890 00000CBB F606[A004]80                     test    byte [multrk_flag], 80h ; multrk_on
15891 00000CC0 7530                             jnz     short disk_ret
15892                                   its_removable:
15893 00000CC2 80E13F                           and     cl, 3Fh         ; eliminate cylinder bits from sector
15894 00000CC5 30E4                             xor     ah, ah
15895 00000CC7 2906[2201]                       sub     [seccnt], ax    ; reduce count of sectors to go      next sector
15896 00000CCB 00C1                             add     cl, al
15897 00000CCD 880E[3101]                       mov     [cursec], cl
15898 00000CD1 263A4D13                         cmp     cl, [es:di+13h]         ; [es:di+BDS.secpertrack]
15899                                                                   ; see if sector/track limit reached
15900 00000CD5 761B                             jbe     short disk_ret
15901 00000CD7 C606[3101]01                     mov     byte [cursec], 1 ; start with first sector of next track
15902 00000CDC 8A36[3201]                       mov     dh, [curhd]
15903 00000CE0 FEC6                             inc     dh
15904 00000CE2 263A7515                         cmp     dh, [es:di+15h]         ; [es:di+BDS.heads]
15905 00000CE6 7206                             jb      short noxor
15906 00000CE8 30F6                             xor     dh, dh
15907 00000CEA FF06[3301]                       inc     word [curtrk]
15908                                   noxor:
15909 00000CEE 8836[3201]                       mov     [curhd], dh
15910                                   disk_ret:
15911 00000CF2 F8                               clc
15912 00000CF3 C3                               retn
15913                                   ; ---------------------------------------------------------------------------
15914
15915                                   ; 15/10/2022
15916
15917                                   ; 24/12/2023 - Retro DOS v5.0
15918
15919                                   ;--------------------------------------------------------------
15920                                   ; the request is for write. determine if we are talking about
15921                                   ; the same track and drive. if so, use the fast speed.
15922                                   ;--------------------------------------------------------------
15923
15924                                   do_write:
15925 00000CF4 3A16[7600]                       cmp     dl, [step_drv]
15926 00000CF8 7506                             jnz     short do_norm  ; we have changed drives
15927                                           ; 24/12/2023
15928 00000CFA 263A6D78                         cmp     ch, [es:di+78h]
15929                                           ;cmp    ch, [es:di+46h]         ; [es:di+BDS.track]
15930 00000CFE 7494                             jz      short do_fast  ; we are still on the same track
15931                                   do_norm:
15932 00000D00 E87500                           call    normspeed
15933 00000D03 EB92                             jmp     short testerr
15934                                   ; ---------------------------------------------------------------------------
15935
15936                                   ;--------------------------------------------------------------
15937                                   ; we have a verify request also. get state info and go verify
15938                                   ;--------------------------------------------------------------
15939
15940                                   doverify:
15941 00000D05 58                               pop     ax
15942 00000D06 50                               push    ax
15943 00000D07 B404                             mov     ah, 4
15944 00000D09 E89000                           call    fastspeed
15945 00000D0C 73A5                             jnb     short noverify
15946
15947                                   ; check the error returned in ah to see if it is a soft ecc error.
15948                                   ; if it is not we needn't do anything special. if it is a soft
15949                                   ; ecc error then decrement the soft_ecc_cnt error retry count. if
15950                                   ; this retry count becomes 0 then we just ignore the error and go to
15951                                   ; no_verify but if we can still try then we call the routine to reset
15952                                   ; the disk and go to dskerr1 to retry the operation.
15953
15954 00000D0E 80FC11                           cmp     ah, 11h         ; soft ecc error ?
15955 00000D11 750B                             jnz     short not_softecc_err
15956 00000D13 FF0E[A504]                       dec     word [soft_ecc_cnt]
15957 00000D17 749A                             jz      short noverify ; no more retry
15958 00000D19 E81F06                           call    ResetDisk       ; reset disk
15959 00000D1C EB3E                             jmp     short dskerr1  ; retry
15960                                   ; ---------------------------------------------------------------------------
15961
15962                                   not_softecc_err:                ; other error.
15963 00000D1E E81A06                           call    ResetDisk
15964 00000D21 FF0E[A304]                       dec     word [vretry_cnt]
15965 00000D25 EB1C                             jmp     short dskerr0
15966                                   ; ---------------------------------------------------------------------------
15967
15968                                   ;--------------------------------------------------------------
15969                                   ; need to special case the change-line error ah=06h.
15970                                   ; if we get this, we need to return it.
15971                                   ;--------------------------------------------------------------
15972
15973                                   dskerr:
15974 00000D27 803E[7700]00                     cmp     byte [fhave96], 0       ; do we have changeline       support?
15975 00000D2C 7403                             jz      short dskerr_nochangeline ; brif not
15976 00000D2E E8BE0E                           call    checkio
15977                                   dskerr_nochangeline:
15978 00000D31 803E[A704]01                     cmp     byte [multitrk_format_flag], 1 ; multi trk format request?
15979 00000D36 7508                             jnz     short dochkagain ; no more retry.
15980 00000D38 BD0100                           mov     bp, 1
15981 00000D3B C606[A704]00                     mov     byte [multitrk_format_flag], 0 ; clear the flag.
15982                                   dochkagain:
15983 00000D40 E89E00                           call    again
15984                                   dskerr0:
15985 00000D43 7420                             jz      short harderr
15986                                           ; 24/12/2023
15987 00000D45 26F6453F01                       test    byte [es:di+3Fh], 1
15988                                           ;test   byte [es:di+23h], 1 ; [es:di+BDS.flags]
15989                                                                   ; fnon_removable
15990 00000D4A 7505                             jnz     short skip_timeout_chk
15991 00000D4C 80FC80                           cmp     ah, 80h         ; timeout?
15992 00000D4F 7414                             jz      short harderr
15993                                   skip_timeout_chk:
15994 00000D51 80FCCC                           cmp     ah, 0CCh        ; write fault error?
15995 00000D54 740A                             jz      short write_fault_err ;       then, don't retry.
15996 00000D56 C706[A504]0500                   mov     word [soft_ecc_cnt], 5 ; MAXERR
```

```
                                                        ; set soft_ecc_cnt back      to maxerr
                                dskerr1:
00000D5C 58                             pop     ax              ; restore sector count
                                        ;jmp    retry
                                        ; 09/12/2022
00000D5D E9F1FE                         jmp     _retry
                                ; ---------------------------------------------------------------------------

                                write_fault_err:
00000D60 BD0100                         mov     bp, 1           ; just retry only once
                                                                ; for write fault error.
00000D63 EBF7                           jmp     short dskerr1

                                        ; fall into harderr
                                ; ---------------------------------------------------------------------------

                                ; entry point for routines that call maperror themselves

                                harderr:
00000D65 E84100                         call    maperror
                                harderr2:
00000D68 C606[1E01]FF                   mov     byte [tim_drv], 0FFh
                                                                ; force a media check through rom
00000D6D 8B0E[2201]                     mov     cx, [seccnt]    ; get count of sectors to go
00000D71 8B26[3501]                     mov     sp, [spsav]     ; recover entry       stack pointer

                                ; since we are performing a non-local goto, restore the disk parameters

                                        ;jmp    diddleback
                                        ; 09/12/2022
00000D75 E951FE                         jmp     diddle_back

                                ; =============== S U B   R O U T I N E =======================================

                                ; change settle value from settlecurrent to whatever is appropriate
                                ; note that this routine is never called for a fixed disk.

                                        ; 19/10/2022
                                normspeed:
00000D78 803E[A905]00                   cmp     byte [media_set_for_format], 0
00000D7D 751D                           jnz     short fastspeed
00000D7F 06                             push    es
00000D80 50                             push    ax
00000D81 A0[2801]                       mov     al, [settleslow]
00000D84 C436[2D01]                     les     si, [dpt]       ; current disk parm table
00000D88 26884409                       mov     [es:si+9], al   ; [es:si+DISK_PARMS.DISK_HEAD_STTL]
00000D8C 58                             pop     ax
00000D8D 07                             pop     es
00000D8E E80B00                         call    fastspeed
                                        ; 24/12/2023
                                        ;push   es
                                        ;les    si, [dpt]
                                        ;mov    byte [es:si+9], 1 ; [es:si+DISK_PARMS.DISK_HEAD_STTL]
                                        ;                       ; 1 is fast settle value
                                        ;pop    es
00000D91 1E                             push    ds
00000D92 C536[2D01]                     lds     si, [dpt]
00000D96 C6440901                       mov     byte [si+9], 1
00000D9A 1F                             pop     ds

00000D9B C3                             retn

                                ; =============== S U B   R O U T I N E =======================================

                                ; if the drive has been marked as too big (i.e. starting sector of the
                                ; partition is > 16 bits, then always return drive not ready.

                                        ; 24/12/2023 - Retro DOS v5.0
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0DDDh)
                                fastspeed:
                                        ;;test  byte [es:di+3Bh], 80h ; [es:di+BDS.fatsiz]
                                        ;test   byte [es:di+1Fh], 80h ; [es:di+BDS.fatsiz]
                                        ;                       ; ftoobig
                                        ;jnz    short notready
00000D9C 06                             push    es
00000D9D 8E06[A804]                     mov     es, [xfer_seg]
00000DA1 CD13                           int     13h             ; DISK -
00000DA3 8C06[A804]                     mov     [xfer_seg], es
00000DA7 07                             pop     es
00000DA8 C3                             retn
                                ; ---------------------------------------------------------------------------
                                ;       ; 24/12/2023
                                ;notready:
                                        ;stc
                                        ;mov    ah, 80h
                                        ;retn

                                ; =============== S U B   R O U T I N E =======================================

                                ; map error returned by rom in ah into corresponding code to be returned to
                                ; dos in al. trashes di. guaranteed to set carry.

                                maperror:
00000DA9 51                             push    cx
00000DAA 06                             push    es
00000DAB 1E                             push    ds              ; set es=Bios_Data
00000DAC 07                             pop     es
00000DAD 88E0                           mov     al, ah          ; put error code in al
00000DAF A2[4601]                       mov     [lsterr], al    ; terminate list with error code
                                        ; 24/12/2023
00000DB2 B90B00                         mov     cx, 11 ; PCDOS 7.1 ; 02/09/2023
                                        ;mov    cx, 9           ; numerr (= errout-errin)
                                                                ; number of possible error conditions
00000DB5 BF[3C01]                       mov     di, errin       ; point to error conditions
00000DB8 F2AE                           repne scasb

                                        ; 24/12/2023
                                        ; 02/09/2023
00000DBA 8A450A                         mov     al, [di+10] ; PCDOS 7.1 IBMBIO.COM
                                        ; 10/12/2022
                                        ;mov    al, [di+8]      ; [di+numerr-1]
                                                                ; get translation
                                        ; 19/10/2022 - Temporary !
                                        ;db     8Ah, 85h, 8, 0 ; mov al, [di+8]
00000DBD 07                             pop     es
00000DBE 59                             pop     cx
00000DBF F9                             stc                     ; flag error condition
00000DC0 C3                             retn

                                ; =============== S U B   R O U T I N E =======================================

                                ; set the time of last access for this drive.
                                ; this is done only for removable media. es:di -> bds
```

```
                                set_tim:
00000DC1 50                             push    ax
00000DC2 E86CF7                         call    GetTickCnt      ; Does INT 1A ah=0 & updates daycnt

                                ; we have the new time. if we see that the time has passed,
                                ; then we reset the threshold counter...

                                        ; 24/12/2023 - Retro DOS v5.0
00000DC5 263B5579                       cmp     dx, [es:di+79h]         ; PCDOS 7.1 IBMBIO.COM
                                        ;cmp    dx, [es:di+47h]         ; [es:di+BDS.tim_lo]
00000DC9 7506                           jne     short setaccess
                                        ; 24/12/2023
00000DCB 263B4D7B                       cmp     cx, [es:di+7Bh]         ; PCDOS 7.1 IBMBIO.COM
                                        ;cmp    cx, [es:di+49h]         ; [es:di+BDS.tim_hi]
                                        ;jz     short done_set
                                        ; 12/12/2022
00000DCF 740E                           je      short done_set2
                                setaccess:
00000DD1 C606[1D01]00                   mov     byte [accesscount], 0

                                        ; 24/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
00000DD6 26895579                       mov     [es:di+79h], dx
00000DDA 26894D7B                       mov     [es:di+7Bh], cx
                                        ;mov    [es:di+47h], dx         ; [es:di+BDS.tim_lo]
                                        ;mov    [es:di+49h], cx         ; [es:di+BDS.tim_hi]
                                done_set:
00000DDE F8                             clc
                                done_set2:      ; 12/12/2022
00000DDF 58                             pop     ax
00000DE0 C3                             retn

                                ; =============== S U B   R O U T I N E =======================================

                                ; this routine is called if an error occurs while formatting or verifying.
                                ; it resets the drive,and decrements the retry count.
                                ; on entry - ds:di - points to bds for the drive
                                ;       bp    - contains retry count
                                ; on exit    flags indicate result of decrementing retry count

                                again:
00000DE1 E85705                         call    ResetDisk
00000DE4 80FC06                         cmp     ah, 6
00000DE7 7402                           jz      short dont_dec_retry_count ; If      it is a media change error
                                                                        ; do not decrement retry count.
00000DE9 4D                             dec     bp              ; decrement retry count
00000DEA C3                             retn
                                ; ---------------------------------------------------------------------------


                                dont_dec_retry_count:
00000DEB 08E4                           or      ah, ah
00000DED C3                             retn

                                ;---------------------------------------------------------------------------
                                ; Retro DOS v5.0 - PCDOS 7.1 IBMBIO.COM - BIOSCODE:0E30h
                                ;---------------------------------------------------------------------------
                                ; 24/12/2023 - Retro DOS v5.0
                                ;;;;

00000DEE 00                     ioctl_drvnum:       db 0

                                        ; 24/12/2023

                                ; =============== S U B R O U T I N E =======================================


                                get_phy_drv_num:
00000DEF E8B2F7                         call    SetDrive        ; get physical drive number
                                                                ; INPUT: al = logical drive number (BDS.drivelet)
                                                                ; OUTPUT: physical drive number (BDS.drivenum)
00000DF2 268A5504                       mov     dl, [es:di+4]   ; [es:di+BDS.drivenum]
00000DF6 C3                             retn

                                ; =============== S U B R O U T I N E =======================================


                                        ; 24/12/2023
                                ioctl_output:
00000DF7 E8F5FF                         call    get_phy_drv_num
00000DFA 2E8816[EE0D]                   mov     [cs:ioctl_drvnum], dl
00000DFF B441                           mov     ah, 41h
00000E01 BBAA55                         mov     bx, 55AAh
00000E04 CD13                           int     13h             ; DISK - Check for INT 13h Extensions
                                                                ; BX = 55AAh, DL = drive number
                                                                ; Return: CF set if not supported
                                                                ; AH = extensions version
                                                                ; BX = AA55h
                                                                ; CX = Interface support bit map
00000E06 7235                           jc      short int13h_exts_err
                                ioctl_input_1:
00000E08 C43E[1200]                     les     di, [ptrsav]
00000E0C 26C47D0E                       les     di, [es:di+14] ; [es:di+IOCTL_REQ.MINORFUNCTION]
00000E10 723E                           jc      short ioctl_input_2
00000E12 B80046                         mov     ax, 4600h       ; Eject removable media
00000E15 263805                         cmp     [es:di], al     ; al = 0 ; disk ioctl function = 0
00000E18 7417                           je      short ioctl_output_1
00000E1A 26803D01                       cmp     byte [es:di], 1 ; al = 1 ; disk ioctl function = 1
00000E1E 751B                           jne     short ioctl_output_2
00000E20 B80145                         mov     ax, 4501h       ; Lock/unlock media
                                                                ; (al, 0 = lock, 1 = unlock)
00000E23 26807D0100                     cmp     byte [es:di+1], 0 ; unlock (reverse of INT 13h ah=45h)
00000E28 7407                           jz      short ioctl_output_1
00000E2A 26384501                       cmp     [es:di+1], al  ; lock (reverse of INT 13h ah=45h)
00000E2E 750B                           jne     short ioctl_output_2
00000E30 48                             dec     ax
                                ioctl_output_1:
00000E31 2E8A16[EE0D]                   mov     dl, [cs:ioctl_drvnum]
00000E36 CD13                           int     13h             ; DISK - IBM/MS Extension - LOCK/UNLOCK DRIVE (DL - drive, [SI - disk address
packet)
00000E38 7203                           jc      short int13h_exts_err
                                ioctl_lock_err:
                                        ; cf=0
                                ioctl_output_ret:
                                        ;clc
00000E3A C3                             retn
                                ; ---------------------------------------------------------------------------

                                ioctl_output_2:
00000E3B B401                           mov     ah, 1
                                int13h_exts_err:
00000E3D 80FCB0                         cmp     ah, 0B0h        ; volume not locked in drive
00000E40 74F8                           je      short ioctl_lock_err
00000E42 80FCB4                         cmp     ah, 0B4h        ; lock count exceeded
00000E45 74F3                           je      short ioctl_lock_err
00000E47 E9DAF7                         jmp     err_exitj

                                ; =============== S U B R O U T I N E =======================================
```

```
                                        ; 24/12/2023
                                ioctl_input:
00000E4A E8A2FF                         call    get_phy_drv_num
00000E4D F9                             stc
00000E4E EBB8                           jmp     short ioctl_input_1
                                ioctl_input_2:
00000E50 26803D06                       cmp     byte [es:di], 6       ; disk ioctl function = 6
00000E54 75E5                           jne     short ioctl_output_2
00000E56 B80245                         mov     ax, 4502h       ; get lock status
00000E59 CD13                           int     13h             ; DISK - IBM/MS Extension - LOCK/UNLOCK DRIVE (DL - drive, [SI - disk address
packet)
00000E5B 72E0                           jc      short int13h_exts_err
00000E5D BB0C00                         mov     bx, 0Ch         ; bit 1 lock bit
00000E60 3C00                           cmp     al, 0           ; not locked
00000E62 7402                           jz      short ioctl_input_3
00000E64 B30E                           mov     bl, 0Eh
                                ioctl_input_3:
00000E66 53                             push    bx
00000E67 B404                           mov     ah, 4
00000E69 B90101                         mov     cx, 101h
00000E6C B601                           mov     dh, 1
00000E6E CD13                           int     13h             ; DISK - VERIFY SECTORS
                                                                ; AL = number of sectors to verify, CH = track, CL = sector
                                                                ; DH = head, DL = drive
                                                                ; Return: CF set on error, AH = status
                                                                ; AL = number of sectors verified
00000E70 5B                             pop     bx
00000E71 80FC31                         cmp     ah, 31h         ; no media in drive (IBM/MS INT 13 extensions)
00000E74 740B                           je      short ioctl_input_5
00000E76 80FC80                         cmp     ah, 80h         ; timeout (not ready)
00000E79 7406                           je      short ioctl_input_5
                                ioctl_input_4:
00000E7B 26895D01                       mov     [es:di+1], bx
00000E7F EBB9                           jmp     short ioctl_lock_err
                                ioctl_input_5:
00000E81 81CB0108                       or      bx, 801h        ; bit 0 error bit (1 = error, 31h or 80h)
                                                                ; bit 11 (not ready -removable media error- bit)
                                                                ; if bit 11 = 0, another error (except 31h and 80h)
00000E85 EBF4                           jmp     short ioctl_input_4

                                ; ----------------------------------------------------------------------------
                                ;;;;

                                ; 16/10/2022 - Retro DOS 4.0 (Modified MSDOS 5.0 IO.SYS)

                                ;----------------------------------------------------------------------------
                                ; MSDIOCTL.ASM - MSDOS 6.0 - 1991
                                ;----------------------------------------------------------------------------
                                ; 11/03/2019 - Retro DOS v4.0

                                ; 18/03/2019

                                ; ============================================================================
                                ;
                                ; NOTE: GetAccessFlag/SetAccessFlag is unpublished function.
                                ;
                                ;      This function is intended to give the user to control the
                                ;      bds table flags of unformatted_media bit.
                                ;      GetAccessFlag will show the status -
                                ;    a_DiskAccess_Control.dac_access_flag = 0 disk i/o not allowed
                                ;                                           1 disk i/o allowed
                                ;      SetAccessFlag will set/reset the unformatted_media bit in flags -
                                ;    a_DiskAccess_Control.dac_access_flag = 0 allow disk i/o
                                ;                                           1 disallow disk i/o
                                ; ============================================================================

                                        ; generic ioctl dispatch tables

                                ; BIOSCODE:0C3Ch (MSDOS 6.21, IO.SYS)

                                ; 24/12/2023
                                ; BIOSCODE:0ECAh (PCDOS 7.1, IBMBIO.COM)

                                ; ----------------------------------------------------------------------------
                                        ; 24/12/2023
                                        ;db 0
                                ; 09/12/2022
                                %if 0

                                IoReadJumpTable: db 8      ; ((IoWriteJumpTable-IoReadJumpTable)-1)/2
                                        dw 0CA7h        ; 60h   ; GetDeviceParameters
                                        dw 0EE8h        ; 61h   ; ReadTrack
                                        dw 0E86h        ; 62h   ; VerifyTrack
                                        dw 0CA3h                ; Cmd_Error_Proc
                                        dw 0CA3h                ; Cmd_Error_Proc
                                        dw 0CA3h                ; Cmd_Error_Proc
                                        dw 119Ah        ; 66h   ; GetMediaId
                                        dw 1269h        ; 67h   ; GetAccessFlag ; unpublished function
                                        dw 12C1h        ; 68h   ; SenseMediaType

                                IoWriteJumpTable: db 7     ; ((IOC_DC_Table-IoWriteJumpTable)-1)/2
                                        dw 0CF3h        ; 40h   ; SetDeviceParameters
                                        dw 0EEFh        ; 41h   ; WriteTrack
                                        dw 0DC1h        ; 42h   ; FormatTrack
                                        dw 0CA3h                ; Cmd_Error_Proc
                                        dw 0CA3h                ; Cmd_Error_Proc
                                        dw 0CA3h                ; Cmd_Error_Proc
                                        dw 11D2h        ; 46h   ; SetMediaId
                                        dw 1280h        ; 47h   ; SetAccessFlag ; unpublished function

                                %endif
                                        ; 24/12/2023 - Retro DOS v5.0
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0ECAh)

                                        ; 09/12/2022
                                IoReadJumpTable:
00000E87 10                             db ((IoWriteJumpTable-IoReadJumpTable)-1)/2 ; 15
00000E88 [1A0F]                         dw GetDeviceParameters; 60h
00000E8A [9311]                         dw ReadTrack          ; 61h
00000E8C [3311]                         dw VerifyTrack        ; 62h
00000E8E [160F]                         dw Cmd_Error_Proc
00000E90 [160F]                         dw Cmd_Error_Proc
00000E92 [160F]                         dw Cmd_Error_Proc
00000E94 [1214]                         dw GetMediaId         ; 66h
00000E96 [EF14]                         dw GetAccessFlag      ; 67h ; unpublished function
00000E98 [4415]                         dw SenseMediaType     ; 68h
                                        ; 24/12/2023
                                        ; Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
00000E9A [160F]                         dw Cmd_Error_Proc     ; 69h
00000E9C [160F]                         dw Cmd_Error_Proc     ; 6Ah
00000E9E [160F]                         dw Cmd_Error_Proc
00000EA0 [160F]                         dw Cmd_Error_Proc
00000EA2 [160F]                         dw Cmd_Error_Proc
```

```
16367 00000EA4 [160F]                    dw Cmd_Error_Proc     ; 6Eh
16368 00000EA6 [C815]                    dw GetDrvMapInfo      ; 6Fh
16369
16370                                IoWriteJumpTable:
16371 00000EA8 0A                        db ((IOC_DC_Table-IoWriteJumpTable)-1)/2 ; 9
16372 00000EA9 [7A0F]                    dw SetDeviceParameters; 40h
16373 00000EAB [9A11]                    dw WriteTrack         ; 41h
16374 00000EAD [6D10]                    dw FormatTrack        ; 42h
16375 00000EAF [160F]                    dw Cmd_Error_Proc
16376 00000EB1 [160F]                    dw Cmd_Error_Proc
16377 00000EB3 [160F]                    dw Cmd_Error_Proc
16378 00000EB5 [5214]                    dw SetMediaId         ; 46h
16379 00000EB7 [0415]                    dw SetAccessFlag      ; 47h ; unpublished function
16380                                    ; 24/12/2023
16381                                    ; Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
16382 00000EB9 [8115]                    dw SetLockState               ; 48h
16383 00000EBB [9815]                    dw EjectMedia         ; 49h
16384
16385                                ; ============================================================================
16386                                ; IOC_DC_Table
16387                                ;
16388                                ; This table contains all of the valid generic IOCtl Minor codes for
16389                                ; major function 08 to be used by the Ioctl_Support_Query function.
16390                                ; Added for 5.00
16391                                ; ============================================================================
16392
16393                                    ; 24/12/2023 - Retro DOS v5.0
16394                                    ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0F00h)
16395
16396 00000EBD 60                    IOC_DC_Table:      db 60h                ; GET_DEVICE_PARAMETERS
16397 00000EBE 40                        db 40h                ; SET_DEVICE_PARAMETERS
16398 00000EBF 61                        db 61h                ; READ_TRACK
16399 00000EC0 41                        db 41h                ; WRITE_TRACK
16400 00000EC1 62                        db 62h                ; VERIFY_TRACK
16401 00000EC2 42                        db 42h                ; FORMAT_TRACK
16402 00000EC3 66                        db 66h                ; GET_MEDIA_ID
16403 00000EC4 46                        db 46h                ; SET_MEDIA_ID
16404 00000EC5 67                        db 67h                ; GET_ACCESS_FLAG
16405 00000EC6 47                        db 47h                ; SET_ACCESS_FLAG
16406 00000EC7 68                        db 68h                ; SENSE_MEDIA_TYPE
16407                                    ; 24/12/2023
16408                                    ; Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
16409 00000EC8 48                        db 48h                ; SET_LOCK_STATE
16410 00000EC9 49                        db 49h                ; EJECT_MEDIA
16411 00000ECA 6F                        db 6Fh                ; GET_DRV_MAP_INFO
16412
16413                                ;IOC_DC_TABLE_LEN EQU $ - IOC_DC_Table
16414
16415                                    ; 24/12/2023 - Retro DOS v5.0
16416                                    ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0F0Eh)
16417
16418 00000ECB 00                    new_genioctl:      db 0
16419
16420                                ; ----------------------------------------------------------------------------
16421
16422                                ; 16/10/2022
16423
16424                                ; ============================================================================
16425                                ; Do_Generic_IOCtl: perform generic ioctl request
16426                                ;
16427                                ;    input: AL contains logical drive
16428                                ;
16429                                ;   functions are dispatched through a call. On return, carry indicates
16430                                ;   error code in al. Note::bES:b& ds undefined on return from
16431                                ;   subfunctions.
16432                                ;
16433                                ; ============================================================================
16434
16435                                ; 11/03/2019
16436                                    ; 24/12/2023 - Retro DOS v5.0
16437                                    ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0F0Fh)
16438
16439                                    ; 19/10/2022
16440                                do_generic_ioctl:                 ; 2C7h:0C6Bh = 70h:31DBh
16441 00000ECC E8D5F6                      call   SetDrive       ; ES:DI Points to bds for drive
16442
16443                                    ; 24/12/2023
16444                                    ;;;
16445 00000ECF 2EC606[CB0E]00            mov    byte [cs:new_genioctl], 0
16446                                                          ; 0, old generic ioctl function
16447 00000ED5 06                        push   es
16448 00000ED6 C41E[1200]                les    bx, [ptrsav]   ; ES:BX Points to request header
16449 00000EDA 26807F0D08                cmp    byte [es:bx+0Dh], 8 ; [es:bx+IOCTL_REQ.MAJORFUNCTION]
16450                                                          ; RAWIO
16451                                    ; 24/12/2023
16452                                    ;mov    al, [es:bx+0Eh]       ; [es:bx+IOCTL_REQ.MINORFUNCTION]
16453                                    ;pop    es
16454                                    ;jnz    short IoctlFuncErr
16455 00000EDF 740A                      jz     short chk_genioctl_minor
16456 00000EE1 2EFE06[CB0E]              inc    byte [cs:new_genioctl]
16457                                                          ; 1, new generic ioctl function (FAT32)
16458 00000EE6 26807F0D48                cmp    byte [es:bx+0Dh], 48h ; Generic IOCtl Request support
16459                                                  ; (called only if bit 6 of attribute is set to 1)
16460                                chk_genioctl_minor:
16461 00000EEB 268A470E                  mov    al, [es:bx+0Eh]       ; [es:bx+IOCTL_REQ.MINORFUNCTION]
16462 00000EEF 07                        pop    es
16463 00000EF0 7525                      jnz    short IoctlFuncErr
16464                                    ;;;
16465
16466                                    ; cas note: Could do the above two blocks in reverse order.
16467                                    ; Would have to preserve al for SetDrive
16468
16469                                    ; 10/12/2022
16470 00000EF2 BE[870E]                  mov    si, IoReadJumpTable
16471                                    ;mov    si, 0C3Ch      ; IoReadJumpTable
16472                                                          ; at 2C7h:0C3Ch       = 70h:31ACh
16473 00000EF5 A820                      test   al, 20h        ; GEN_IOCTL_FN_TST ; test of req. function
16474 00000EF7 7503                      jnz    short NotGenericWrite ; function is a read.
16475                                    ; 10/12/2022
16476 00000EF9 BE[A80E]                  mov    si, IoWriteJumpTable
16477                                    ;mov    si, 0C4Fh      ; IoWriteJumpTable
16478                                                          ; at 2C7h:0C4Fh       = 70h:31BFh
16479                                NotGenericWrite:
16480 00000EFC 24DF                      and    al, 0DFh       ; ~GEN_IOCTL_FN_TST ; get rid of read/write bit
16481 00000EFE 2C40                      sub    al, 40h        ; offset for base function
16482 00000F00 2E3A04                    cmp    al, [cs:si]
16483 00000F03 7712                      ja     short IoctlFuncErr
16484 00000F05 98                        cbw
16485                                    ; 24/12/2023
16486                                    ;shl    ax, 1
16487 00000F06 01C0                      add    ax, ax
16488 00000F08 46                        inc    si
16489 00000F09 01C6                      add    si, ax
16490 00000F0B 2EFF14                    call   near [cs:si]
```

```
                                        ;call   word ptr cs:[si]
00000F0E 2E8E1E[3000]                   mov     ds, [cs:BIOSDATAWORD]
                                        ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
                                                                ; 2C7h:30h = 70h:25A0h
00000F13 B481                           mov     ah, 81h         ; Return this status in       case of carry
00000F15 C3                             retn                    ; Pass carry flag through to exit code
                                ; ---------------------------------------------------------------------------

                                        ; Cmd_Error_Proc is called as a procedure and also use
                                        ; as a fall through from above
                                Cmd_Error_Proc:                         ; 2C7h:0CA3h = 70h:3213h
00000F16 5A                             pop     dx
                                IoctlFuncErr:
00000F17 E9BBF1                         jmp     bc_cmderr
                                ; ---------------------------------------------------------------------------

                                ; 16/10/2022

                                ; ===========================================================================
                                ;** GetDeviceParameters:
                                ;
                                ;   GetDeviceParameters implements the generic ioctl function:
                                ;   majorcode=RAWIO, minorcode=GetDeviceParameters (60h)
                                ;
                                ;   ENTRY   (ES:di) = BDS for drive
                                ;           PtrSav = long pointer to request header
                                ;   EXIT    ??? BUGBUG
                                ;   USES    ??? BUGBUG
                                ; ===========================================================================

                                        ; 24/12/2023 - Retro DOS v5.0
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0F5Dh)

                                        ; 19/10/2022
                                GetDeviceParameters:
                                        ; Copy info from bds to the device parameters packet

00000F1A C51E[1200]                     lds     bx, [ptrsav]    ; DS:BX points to request header
00000F1E C55F13                         lds     bx, [bx+19]     ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET]
                                                                ; (DS:BX) = return buffer
                                        ; 24/12/2023
00000F21 268A453E                       mov     al, [es:di+3Eh]
                                        ;mov    al, [es:di+34] ; [es:di+BDS.formfactor]
00000F25 884701                         mov     [bx+1], al      ; [bx+A_DEVICEPARAMETERS.DP_DEVICETYPE]
                                        ; 24/12/2023
00000F28 268B453F                       mov     ax, [es:di+3Fh]
                                        ;mov    ax, [es:di+35] ; [es:di+BDS.flags]
00000F2C 83E003                         and     ax, 3           ; fnon_removable+fchangeline
                                                                ; Mask off other bits
00000F2F 894702                         mov     [bx+2], ax      ; [bx+A_DEVICEPARAMETERS.DP_DEVICEATTRIBUTES]
                                        ; 24/12/2023
00000F32 268B4541                       mov     ax, [es:di+41h]
                                        ;mov    ax, [es:di+37] ; [es:di+BDS.cylinders]
00000F36 894704                         mov     [bx+4], ax      ; [bx+A_DEVICEPARAMETERS.DP_CYLINDERS]
00000F39 30C0                           xor     al, al          ; Set media type to default
00000F3B 884706                         mov     [bx+6], al      ; [bx+A_DEVICEPARAMETERS.DP_MEDIATYPE]

                                        ; copy recommended bpb

                                        ; 24/12/2023
00000F3E 8D7543                         lea     si, [di+43h]
                                        ;lea    si, [di+39]     ; [di+BDS.rbytespersec]       = [di+BDS.R_BPB]
00000F41 F60701                         test    byte [bx], 1    ; [bx+A_DEVICEPARAMETERS.DP_SPECIALFUNCTIONS]
                                                                ; BUILD_DEVICE_BPB
00000F44 7412                           jz      short UseBpbPresent
00000F46 1E                             push    ds              ; Save request packet segment
00000F47 2E8E1E[3000]                   mov     ds, [cs:BIOSDATAWORD]
                                        ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
                                                                ; 2C7h:30h = 70h:25A0h
                                                                ; Point back to Bios_Data
00000F4C E828FA                         call    checksingle
00000F4F E884F7                         call    GetBp           ; Build the bpb from scratch
00000F52 1F                             pop     ds              ; Restore request packet segment
00000F53 7224                           jb      short GetParmRet
00000F55 8D7506                         lea     si, [di+6]      ; [di+BDS.bytespersec] = [di+BSD.DP_BPB]
                                                                ; Use this subfield of bds instead
                                UseBpbPresent:
00000F58 8D7F07                         lea     di, [bx+7]      ; [bx+A_DEVICEPARAMETERS.DP_BPB]
                                                                ; This is where       the result goes
                                        ; 24/12/2023
00000F5B 31D2                           xor     dx, dx ; 0

                                        ; 24/12/2023
00000F5D B91F00                         mov     cx, 31          ; A_BPB.size = 31
                                        ;mov    cx, 25          ; A_BPB.size - 6
                                                                ; For now use 'small' bpb
                                        ; 24/12/2023
                                        ;;;
00000F60 2E3816[CB0E]                   cmp     [cs:new_genioctl], dl ; 0 ? ; *
00000F65 7404                           jz      short gdp_1     ; old type (FAT12 & FAT16) structure
                                        ;mov    cx, 53          ; FAT32 BPB size
                                        ;mov    dx, 32          ; 53+32 = 85 bytes (A_BPB_FAT32.size)
00000F67 B135                           mov     cl, 53
00000F69 B220                           mov     dl, 32
                                gdp_1:
                                        ;;;
00000F6B 1E                             push    ds              ; reverse segments for copy
00000F6C 06                             push    es
00000F6D 1F                             pop     ds
00000F6E 07                             pop     es
00000F6F F3A4                           rep movsb

                                        ; 24/12/2023
                                        ;;;
00000F71 89D1                           mov     cx, dx          ; 0 or 32
00000F73 E304                           jcxz    gdp_2
00000F75 30C0                           xor     al, al          ; 32 zeros
00000F77 F3AA                           rep stosb
                                gdp_2:
                                        ;clc    ; cf is already 0 ; * ; 24/12/2023
                                        ;;;

                                        ; 12/12/2022
                                        ; cf=0 (cmp instruction -above- resets cf)
                                        ;clc
                                GetParmRet:
00000F79 C3                             retn
                                ; ---------------------------------------------------------------------------

                                ; 17/10/2022
                                ; 16/10/2022

                                ; ===========================================================================
                                ; SetDeviceParameters:
```

```
16615                                   ;
16616                                   ; input: ES:di points to bds for drive
16617                                   ; ==========================================================================
16618
16619                                           ; 24/12/2023 - Retro DOS v5.0
16620                                           ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0FC0h)
16621
16622                                           ; 19/10/2022
16623                                   SetDeviceParameters:                    ; 2C7h:0CF3h = 70h:3263h
16624 00000F7A C51E[1200]                         lds     bx, [ptrsav]    ; DS:BX points to request header
16625 00000F7E C55F13                             lds     bx, [bx+19]     ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET]
16626                                           ; 24/12/2023
16627 00000F81 26814D3F4001                       or      word [es:di+3Fh], 140h
16628                                           ;or     word [es:di+23h], 140h ; [es:di+BDS.flags]
16629                                                                   ; fchanged_by_format|fchanged
16630 00000F87 F60702                             test    byte [bx], 2    ; [bx+A_DEVICEPARAMETERS.DP_SPECIALFUNCTIONS]
16631                                                                   ; ONLY_SET_TRACKLAYOUT
16632                                           ;jnz    short setTrackTable
16633                                           ; 24/12/2023
16634 00000F8A 7403                               jz      short sdp_1
16635 00000F8C E98000                             jmp     setTrackTable
16636                                   sdp_1:
16637 00000F8F 8A4701                             mov     al, [bx+1]      ; [bx+A_DEVICEPARAMETERS.DP_DEVICETYPE]
16638                                           ; 24/12/2023
16639 00000F92 2688453E                           mov     [es:di+3Eh], al
16640                                           ;mov    [es:di+34], al ; [es:di+BDS.formfactor]
16641 00000F96 8B4704                             mov     ax, [bx+4]      ; [bx+A_DEVICEPARAMETERS.DP_CYLINDERS]
16642                                           ; 24/12/2023
16643 00000F99 26894541                           mov     [es:di+41h], ax
16644                                           ;mov    [es:di+37], ax ; [es:di+BDS.cylinders]
16645 00000F9D 8B4702                             mov     ax, [bx+2]      ; [bx+A_DEVICEPARAMETERS.DP_DEVICEATTRIBUTES]
16646 00000FA0 1E                                 push    ds
16647                                           ; 17/10/2022
16648 00000FA1 2E8E1E[3000]                       mov     ds, [cs:BIOSDATAWORD]
16649                                           ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16650                                                                   ; 2C7h:30h = 70h:25A0h
16651                                           ;cmp    byte [fhave96], 0
16652 00000FA6 803E[7700]00                       cmp     byte [fhave96], 0
16653 00000FAB 1F                                 pop     ds
16654 00000FAC 7502                               jnz     short HaveChange ; we have changeline support
16655                                           ; 10/12/2022
16656 00000FAE 24FD                               and     al, 0FDh
16657                                           ;and    ax, 0FFFDh      ; ~fchangeline
16658
16659                                           ; Ignore all bits except non_removable and changeline
16660                                   HaveChange:
16661 00000FB0 83E003                             and     ax, 3           ; fnon_removable|fchangeline
16662                                           ; 24/12/2023
16663 00000FB3 268B4D3F                           mov     cx, [es:di+3Fh]
16664                                           ;mov    cx, [es:di+35] ; [es:di+BDS.flags]
16665 00000FB7 81E1F4FD                           and     cx, 0FDF4h      ; ~(fnon_removable|fchangeline|good_tracklayout|unformatted_media)
16666 00000FBB 09C8                               or      ax, cx
16667                                           ; 24/12/2023
16668 00000FBD 2689453F                           mov     [es:di+3Fh], ax
16669                                           ;mov    [es:di+35], ax ; [es:di+BDS.flags]
16670 00000FC1 8A4706                             mov     al, [bx+6]      ; [bx+A_DEVICEPARAMETERS.DP_MEDIATYPE]
16671                                                                   ; Set media type
16672 00000FC4 1E                                 push    ds
16673 00000FC5 2E8E1E[3000]                       mov     ds, [cs:BIOSDATAWORD]
16674                                           ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16675 00000FCA A2[A805]                           mov     [mediatype], al
16676                                           ;mov    ds:mediatype, al
16677
16678                                           ; 24/12/2023
16679                                           ;;;
16680 00000FCD B93500                             mov     cx, 53          ; FAT32 BPB size
16681 00000FD0 2E803E[CB0E]00                     cmp     byte [cs:new_genioctl], 0
16682 00000FD6 7502                               jnz     short sdp_2     ; new type (FAT32) structure
16683                                           ;mov    cx, 31          ; A_BPB.size = 31
16684 00000FD8 B11F                               mov     cl, 31
16685                                   sdp_2:
16686                                           ;;;
16687 00000FDA 1F                                 pop     ds
16688
16689                                           ; The media changed (maybe) so we will have to do a set dasd
16690                                           ; the next time we format a track
16691
16692                                           ; 24/12/2023
16693 00000FDB 26804D3F80                         or      byte [es:di+3Fh], 80h
16694                                           ; 10/12/2022
16695                                           ;or     byte [es:di+35], 80h
16696                                           ;;or    word [es:di+35], 80h ; [es:di+BDS.flags]
16697                                                                   ; set_dasd_true
16698 00000FE0 57                                 push    di              ; Save bds pointer
16699
16700                                           ; Figure out what we are supposed to do with the bpb
16701                                           ; were we asked to install a fake bpb?
16702
16703 00000FE1 F60701                              test    byte [bx], 1    ; [bx+A_DEVICEPARAMETERS.DP_SPECIALFUNCTIONS]
16704                                                                   ; INSTALL_FAKE_BPB
16705 00000FE4 7511                               jnz     short InstallFakeBpb
16706
16707                                           ; were we returning a fake bpb when asked to build a bpb?
16708
16709                                           ; 24/12/2023
16710 00000FE6 26F6453F04                         test    byte [es:di+3Fh], 4
16711                                           ; 10/12/2022
16712                                           ;test   byte [es:di+35], 4
16713                                           ;;test  word [es:di+35], 4 ; [es:di+BDS.flags]
16714                                                                   ; return_fake_bpb
16715 00000FEB 7405                               jz      short InstallRecommendedBpb
16716
16717                                           ; we were returning a fake bpb but we can stop now
16718
16719                                           ; 24/12/2023
16720 00000FED 2680653FFB                         and     byte [es:di+3Fh], 0FBh
16721                                           ; 10/12/2022
16722                                           ;and    byte [es:di+35], 0FBh
16723                                           ;;and   word [es:di+35], 0FFFBh ; [es:di+BDS.flags]
16724                                                                   ; ~return_fake_bpb
16725                                   InstallRecommendedBpb:
16726                                           ; 24/12/2023
16727                                           ;mov    cx, 31          ; A_BPB.size
16728                                           ;lea    di, [di+27h]    ; [di+BDS.R_BPB] = [di+BDS.rbytespersec]
16729                                           ; cx = 53 or 31
16730 00000FF2 8D7D43                           lea     di, [di+43h]    ; (PCDOS 7.1 IBMBIO.COM)
16731 00000FF5 EB08                             jmp     short CopyTheBpb
16732                                   ; ---------------------------------------------------------------------------
16733
16734                                   InstallFakeBpb:
16735                                           ; 24/12/2023
16736 00000FF7 26804D3F04                       or      byte [es:di+3Fh], 4
16737                                           ; 10/12/2022
16738                                           ;or     byte [es:di+35], 4
```

```
16739                                           ;;or    word [es:di+35], 4 ; byte [es:di+BDS.flags]
16740                                                                   ; return_fake_bpb
16741                                           ; 24/12/2023
16742                                           ; cx = 53 or 31
16743                                           ;mov    cx, 25          ; A_BPB.size - 6
16744                                                                   ; move 'smaller' bpb
16745 00000FFC 8D7D06                           lea     di, [di+6]      ; [es:di+BDS.BPB] = [es:di+BDS.bytespersec]
16746                                   CopyTheBpb:
16747 00000FFF 8D7707                           lea     si, [bx+7]      ; [bx+A_DEVICEPARAMETERS.DP_BPB]
16748 00001002 F3A4                             rep movsb
16749 00001004 1E                               push    ds              ; Save packet segment
16750                                           ; 17/10/2022
16751 00001005 2E8E1E[3000]                     mov     ds, [cs:BIOSDATAWORD]
16752                                           ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16753                                                                   ; Setup for ds -> Bios_Data
16754 0000100A E8DD03                           call    RestoreOldDpt   ; Restore the old Dpt from TempDpt
16755 0000100D 1F                               pop     ds              ; Restore packet segment
16756 0000100E 5F                               pop     di              ; Restore bds pointer
16757                                   setTrackTable:
16758                                           ; 24/12/2023
16759                                           ;mov    cx, [bx+38]     ; [bx+26h]
16760                                           ;;;
16761 0000100F 8B4F5C                           mov     cx, [bx+5Ch]    ; [bx+A_DEVICEPARAMETERS.DP_TRACKTABLEENTRIES]
16762                                                                   ; offset 85+7 (A_BPB.size+7) (FAT32)
16763 00001012 2E803E[CB0E]00                   cmp     byte [cs:new_genioctl], 0
16764 00001018 7503                             jnz     short sdp_3     ; new type (FAT32) structure
16765 0000101A 8B4F26                           mov     cx, [bx+26h]    ; [bx+A_DEVICEPARAMETERS.DP_TRACKTABLEENTRIES]
16766                                                                   ; offset 31+7 (A_BPB.size+7)
16767                                   sdp_3:
16768                                           ;;;
16769
16770 0000101D 1E                               push    ds
16771 0000101E 2E8E1E[3000]                     mov     ds, [cs:BIOSDATAWORD]
16772 00001023 890E[AA04]                       mov     [sectorspertrack], cx
16773 00001027 1F                               pop     ds
16774
16775                                           ; 24/12/2023
16776 00001028 2680653FF7                       and     byte [es:di+3Fh], 0F7h
16777                                           ; 10/12/2022
16778                                           ;and    byte [es:di+35], 0F7h
16779                                           ;;and   word [es:di+35], 0FFF7h ; [es:di+BDS.flags]
16780                                                                   ; ~good_tracklayout
16781 0000102D F60704                           test    byte [bx], 4    ; [bx+A_DEVICEPARAMETERS.DP_SPECIALFUNCTIONS]
16782                                                                   ; TRACKLAYOUT_IS_GOOD
16783 00001030 7405                             jz      short UglyTrackLayOut
16784                                           ; 24/12/2023
16785 00001032 26804D3F08                       or      byte [es:di+3Fh], 8
16786                                           ; 10/12/2022
16787                                           ;or     byte [es:di+35], 8
16788                                           ;;or    word [es:di+35], 8 ; [es:di+BDS.flags]
16789                                                                   ; good_tracklayout
16790                                   UglyTrackLayOut:
16791 00001037 83F93F                           cmp     cx, 63          ; MAX_SECTORS_IN_TRACK
16792 0000103A 772D                             ja      short TooManyPerTrack
16793                                           ;jcxz   short SectorInfoSaved
16794 0000103C E329                             jcxz    SectorInfoSaved       ; 19/10/2022
16795
16796 0000103E BF[AC04]                         mov     di, tracktable
16797
16798                                           ; 24/12/2023
16799                                           ;lea    si, [bx+40]     ; [bx+A_DEVICEPARAMETERS.DP_SECTORTABLE]
16800                                           ;;;
16801 00001041 8D775E                           lea     si, [bx+5Eh]    ; [bx+A_DEVICEPARAMETERS.DP_SECTORTABLE]
16802                                                                   ; offset 85+9 (A_BPB.size+9) (FAT32)
16803 00001044 2E803E[CB0E]00                   cmp     byte [cs:new_genioctl], 0
16804 0000104A 7503                             jnz     short sdp_4     ; new type (FAT32) structure
16805 0000104C 8D7728                           lea     si, [bx+28h]    ; [bx+A_DEVICEPARAMETERS.DP_SECTORTABLE]
16806                                                                   ; offset 31+9 (A_BPB.size+9)
16807                                   sdp_4:
16808                                           ;;;
16809
16810                                           ; 17/10/2022
16811 0000104F 2E8E06[3000]                     mov     es, [cs:BIOSDATAWORD]
16812                                           ;mov    es, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16813                                                                   ; Trash our bds pointer
16814                                   StoreSectorInfo:
16815 00001054 47                               inc     di
16816 00001055 47                               inc     di              ; Skip over cylinder and head
16817 00001056 AD                               lodsw                   ; Get sector id
16818 00001057 AA                               stosb                   ; Copy it
16819 00001058 AD                               lodsw                   ; Get sector size
16820
16821                                           ; 24/12/2023
16822                                           ; 02/09/2023 (PCDOS 7.1)
16823                                           ;call   SectSizeToSectIndex
16824                                           ; 18/04/2024
16825                                           ;cmp    ah, 3 ; 02/09/2023
16826 00001059 80FC02                           cmp     ah, 2           ; (0=>128,1=>256,2=>512,3=>1024)
16827                                                                   ; examine upper       byte only
16828 0000105C 7704                             ja      short OneK
16829 0000105E 88E0                             mov     al, ah          ; value in AH is the index!
16830 00001060 EB02                             jmp     short sdp_s
16831                                   OneK:
16832 00001062 B003                             mov     al, 3           ; 1024 bytes per sector
16833                                   sdp_s:
16834 00001064 AA                               stosb                   ; Store sector SIZE index
16835 00001065 E2ED                             loop    StoreSectorInfo
16836                                   SectorInfoSaved:
16837 00001067 F8                               clc
16838 00001068 C3                               retn
16839                                   ; ---------------------------------------------------------------------------
16840
16841                                   TooManyPerTrack:
16842 00001069 B00C                             mov     al, 0Ch
16843 0000106B F9                               stc
16844 0000106C C3                               retn
16845                                   ; ---------------------------------------------------------------------------
16846
16847                                   ; 16/10/2022
16848
16849                                   ; ============================================================================
16850                                   ; FormatTrack:
16851                                   ; if specialfunction byte is 1,then this is a status call to see if there is
16852                                   ; rom support for the combination of sec/trk and # of cyln,and if the
16853                                   ; combination is legal. if specialfunction byte is 0,then format the track.
16854                                   ;
16855                                   ; input: ES:di points to bds for drive
16856                                   ;
16857                                   ; output:
16858                                   ;   for status call:
16859                                   ;   specialfunction byte set to:
16860                                   ;           0 - rom support + legal combination
16861                                   ;           1 - no rom support
16862                                   ;           2 - illegal combination
```

```
16863                                   ;           3 - no media present
16864                                   ;   carry cleared.
16865                                   ;
16866                                   ;   for format track:
16867                                   ;           carry set if error
16868                                   ;
16869                                   ; ==========================================================================
16870
16871                                   ; 16/03/2019
16872                                               ; 24/12/2023 - Retro DOS 5.0
16873                                               ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:10B7h)
16874
16875                                               ; 19/10/2022
16876                                   FormatTrack:
16877 0000106D C51E[1200]                           lds     bx, [ptrsav]
16878 00001071 C55F13                               lds     bx, [bx+19]     ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET
16879 00001074 F60701                               test    byte [bx], 1    ; [bx+A_DEVICEPARAMETERS.DP_SPECIALFUNCTIONS]
16880                                                                       ; STATUS_FOR_FORMAT
16881 00001077 740E                                 jz      short DoFormatTrack
16882 00001079 1E                                   push    ds
16883                                               ; 17/10/2022
16884 0000107A 2E8E1E[3000]                         mov     ds, [cs:BIOSDATAWORD]
16885                                               ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16886 0000107F E82502                               call    SetMediaForFormat ; Also moves current Dpt to TempDpt
16887 00001082 1F                                   pop     ds
16888 00001083 8807                                 mov     [bx], al        ; [bx+A_DEVICEPARAMETERS.DP_SPECIALFUNCTIONS]
16889 00001085 F8                                   clc
16890 00001086 C3                                   retn
16891                                   ; ---------------------------------------------------------------------------
16892
16893                                   DoFormatTrack:
16894                                               ; 24/12/2023 - Retro DOS 5.0
16895 00001087 26807D3E05                           cmp     byte [es:di+3Eh], 5
16896                                               ;cmp    byte [es:di+34], 5 ; [es:di+BDS.formfactor]
16897                                                                       ; DEV_HARDDISK
16898 0000108C 7508                                 jnz     short DoFormatDiskette
16899                                               ; 17/10/2022
16900 0000108E 2E8E1E[3000]                         mov     ds, [cs:BIOSDATAWORD]
16901                                               ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16902                                                                       ; Point to Bios_Data (at 2C7h:30h or 70h:25A0h)
16903 00001093 E99D00                               jmp     VerifyTrack
16904                                   ; ---------------------------------------------------------------------------
16905
16906                                   DoFormatDiskette:
16907 00001096 8B4F01                               mov     cx, [bx+1]
16908 00001099 8B5703                               mov     dx, [bx+3]
16909 0000109C F60702                               test    byte [bx], 2
16910                                               ; 17/10/2022
16911 0000109F 2E8E1E[3000]                         mov     ds, [cs:BIOSDATAWORD]
16912                                               ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16913                                                                       ; Setup ds-> Bios_Data for verify
16914 000010A4 7403                                 jz      short DoFormatDiskette_1
16915 000010A6 E9E500                               jmp     VerifyTrack_Err
16916                                   ; ---------------------------------------------------------------------------
16917
16918                                   DoFormatDiskette_1:
16919 000010A9 E8FB01                               call    SetMediaForFormat ; Also moves current Dpt to TempDpt
16920 000010AC 3C01                                 cmp     al, 1           ;  ROM support for sec/trk,# trks comb?
16921 000010AE 7406                                 jz      short NeedToSetDasd ; Old rom
16922 000010B0 3C03                                 cmp     al, 3           ; Time out error?
16923 000010B2 7507                                 jnz     short NoSetDasd       ; No,fine. (at this point, don't care
16924                                                                       ; about the illegal combination)
16925 000010B4 EB68                                 jmp     short FormatFailed
16926                                   ; ---------------------------------------------------------------------------
16927
16928                                   NeedToSetDasd:
16929 000010B6 52                                   push    dx
16930 000010B7 E89001                               call    SetDasd         ; INT 13h, AH=17h
16931 000010BA 5A                                   pop     dx
16932                                   NoSetDasd:
16933 000010BB E8B9F8                               call    checksingle     ; Do any needed       diskette swapping
16934 000010BE 89D0                                 mov     ax, dx          ; Get track from packet
16935 000010C0 A3[3901]                             mov     [trknum], ax
16936 000010C3 880E[3801]                           mov     [hdnum], cl     ; Store head from packet
16937 000010C7 88CC                                 mov     ah, cl
16938 000010C9 BB[AC04]                             mov     bx, tracktable
16939 000010CC 8B0E[AA04]                           mov     cx, [sectorspertrack]
16940                                               ; 24/12/2023 - Retro DOS 5.0
16941 000010D0 E307                                 jcxz    set_fmt_retry_count
16942                                   StoreCylinderHead:
16943 000010D2 8907                                 mov     [bx], ax        ; Store into TrackTable
16944 000010D4 83C304                               add     bx, 4           ; Skip to next sector field
16945 000010D7 E2F9                                 loop    StoreCylinderHead
16946                                   set_fmt_retry_count:        ; 24/12/2023
16947                                               ;mov    cx, 5           ; MAXERR - Set up retry       count
16948                                               ; 02/09/2023
16949 000010D9 B105                                 mov     cl, 5
16950                                   FormatRetry:
16951 000010DB 51                                   push    cx
16952 000010DC BB[AC04]                             mov     bx, tracktable
16953 000010DF A0[AA04]                             mov     al, [sectorspertrack]
16954 000010E2 B405                                 mov     ah, 5           ; romformat
16955 000010E4 8C1E[A804]                           mov     [xfer_seg], ds
16956 000010E8 E86602                               call    ToRom
16957 000010EB 59                                   pop     cx
16958 000010EC 7216                                 jb      short FormatError
16959 000010EE 51                                   push    cx              ; Now verify the sectors just formatted.
16960                                                                       ; NOTE: because of bug in some BIOSes we have to
16961                                                                       ;       set ES:BX to 00:00
16962 000010EF 53                                   push    bx
16963 000010F0 31DB                                 xor     bx, bx
16964 000010F2 891E[A804]                           mov     [xfer_seg], bx
16965 000010F6 A0[AA04]                             mov     al, [sectorspertrack]
16966 000010F9 B404                                 mov     ah, 4           ; romverify
16967 000010FB B101                                 mov     cl, 1
16968 000010FD E85102                               call    ToRom
16969 00001100 5B                                   pop     bx
16970 00001101 59                                   pop     cx
16971 00001102 7329                                 jnb     short FormatOk
16972                                   FormatError:
16973 00001104 E83402                               call    ResetDisk
16974 00001107 C606[AA05]01                         mov     byte [had_format_error], 1
16975 0000110C 50                                   push    ax
16976 0000110D 51                                   push    cx
16977 0000110E 52                                   push    dx
16978 0000110F E89501                               call    SetMediaForFormat
16979 00001112 3C01                                 cmp     al, 1
16980 00001114 7503                                 jnz     short WhileErr
16981 00001116 E83101                               call    SetDasd
16982                                   WhileErr:
16983 00001119 5A                                   pop     dx
16984 0000111A 59                                   pop     cx
16985 0000111B 58                                   pop     ax
16986 0000111C E2BD                                 loop    FormatRetry
```

```
16987                                   FormatFailed:
16988 0000111E C606[AA05]01                       mov     byte [had_format_error], 1
16989                                                             ; Set the format error flag
16990 00001123 80FC06                             cmp     ah, 6           ; DSK_CHANGELINE_ERR - convert change line
16991 00001126 7502                               jnz     short DoMapIt   ; Error to time out error
16992 00001128 B480                               mov     ah, 80h         ; DSK_TIMEOUT_ERR
16993                                   DoMapIt:
16994 0000112A E97CFC                             jmp     maperror
16995                                   ; ---------------------------------------------------------------------------
16996
16997                                   FormatOk:
16998 0000112D C606[AA05]00                       mov     byte [had_format_error], 0 ; reset the format error flag
16999 00001132 C3                                 retn
17000                                   ; ---------------------------------------------------------------------------
17001
17002                                   ; 16/10/2022
17003
17004                                   ; ============================================================================
17005                                   ;
17006                                   ; VerifyTrack:
17007                                   ;
17008                                   ; input: ES:di points to bds for drive
17009                                   ; ============================================================================
17010
17011                                             ; 24/12/2023 - Retro DOS 5.0
17012                                   VerifyTrack:
17013 00001133 1E                                 push    ds
17014 00001134 C51E[1200]                         lds     bx, [ptrsav]    ; DS:BX points to request header.
17015 00001138 C55F13                             lds     bx, [bx+19]     ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET]
17016
17017                                             ; Come here with DS:[BX] -> packet, ES:[DI] -> bds
17018
17019 0000113B 8B4F03                             mov     cx, [bx+3]      ; [bx+A_VERIFYPACKET.VP_CYLINDER]
17020 0000113E 8B4701                             mov     ax, [bx+1]      ; [bx+A_VERIFYPACKET.VP_HEAD]
17021 00001141 8B5705                             mov     dx, [bx+5]      ; [bx+A_FORMATPACKET.FP_TRACKCOUNT]
17022 00001144 8A1F                               mov     bl, [bx]        ; [bx+A_FORMATPACKET.FP_SPECIALFUNCTIONS]
17023                                                             ; Get option flag word
17024 00001146 1F                                 pop     ds
17025 00001147 C606[2001]04                       mov     byte [rflag], 4       ; romverify
17026 0000114C 890E[3301]                         mov     [curtrk], cx
17027 00001150 A2[3201]                           mov     [curhd], al     ; ASSUME heads < 256
17028 00001153 8B0E[AA04]                         mov     cx, [sectorspertrack]
17029
17030                                             ; Check specialfunctions to see if DO_FAST_FORMAT has been
17031                                             ; specified if not we should go to the normal track verification
17032                                             ; routine. If fast format has been specified we should get the
17033                                             ; number of tracks to be verified and check it to see if it is
17034                                             ; > 255. If it is then it is an error and we should go to
17035                                             ; VerifyTrack_Err. If not multiply the number of tracks by the
17036                                             ; sectors per track to get the total number of sectors to be
17037                                             ; verified. This should also be less than equal to 255
17038                                             ; otherwise we go to same error exit. If everything is okay
17039                                             ; we initialise cx to the total sectors. use ax as a temporary
17040                                             ; register.
17041
17042                                                             ; Special function requested?
17043 00001157 F6C302                             test    bl, 2           ; DO_FAST_FORMAT
17044 0000115A 7421                               jz      short NormVerifyTrack
17045 0000115C 89D0                               mov     ax, dx          ; Get ax = number of trks to verify
17046 0000115E 08E4                               or      ah, ah
17047 00001160 752C                               jnz     short VerifyTrack_Err ; #tracks > 255
17048 00001162 F6E1                               mul     cl
17049 00001164 08E4                               or      ah, ah
17050 00001166 7526                               jnz     short VerifyTrack_Err ; #sectors > 255
17051 00001168 89C1                               mov     cx, ax
17052                                             ; 24/12/2023
17053 0000116A 26F6453F01                         test    byte [es:di+3Fh], 1 ; PCDOS 7.1 IBMBIO.COM
17054                                             ; 10/12/2022
17055                                             ;test   byte [es:di+35], 1
17056                                             ;;test  word [es:di+35], 1 ; [es:di+BDS.flags]
17057                                                             ; fnon_removable
17058 0000116F 740C                               jz      short NormVerifyTrack
17059                                                             ; Multitrack operation = on?
17060                                             ; 10/12/2022
17061                                             ; 19/10/2022
17062 00001171 F606[A004]80                       test    byte [multrk_flag], 80h
17063                                             ;test   word [multrk_flag], 80h ; MULTI_TRK_ON
17064                                             ;;test  ds:multrk_flag,       80h ; MULTI_TRK_ON
17065 00001176 7405                               jz      short NormVerifyTrack
17066 00001178 C606[A704]01                       mov     byte [multitrk_format_flag], 1
17067                                   NormVerifyTrack:
17068 0000117D 31C0                               xor     ax, ax          ; 1st sector
17069 0000117F 31DB                               xor     bx, bx
17070 00001181 891E[A804]                         mov     [xfer_seg], bx ; Use 0:0 as the transfer address for verify
17071 00001185 E83F00                             call    TrackIo
17072 00001188 C606[A704]00                       mov     byte [multitrk_format_flag], 0
17073 0000118D C3                                 retn
17074                                   ; ---------------------------------------------------------------------------
17075
17076                                   VerifyTrack_Err:
17077 0000118E B401                               mov     ah, 1
17078 00001190 E916FC                             jmp     maperror
17079                                   ; ---------------------------------------------------------------------------
17080
17081                                   ; 16/10/2022
17082
17083                                   ; ============================================================================
17084                                   ;
17085                                   ; ReadTrack:
17086                                   ;
17087                                   ; input: ES:di points to bds for drive
17088                                   ;
17089                                   ; ============================================================================
17090
17091                                   ReadTrack:
17092 00001193 C606[2001]02                       mov     byte [rflag], 2       ; romread
17093 00001198 EB05                               jmp     short ReadWriteTrack
17094                                   ; ---------------------------------------------------------------------------
17095
17096                                   WriteTrack:
17097
17098                                   ; ============================================================================
17099                                   ;
17100                                   ; WriteTrack:
17101                                   ;
17102                                   ; input: ES:di points to bds for drive
17103                                   ;
17104                                   ; ============================================================================
17105
17106 0000119A C606[2001]03                       mov     byte [rflag], 3       ; romwrite
17107
17108                                             ; Fall into ReadWriteTrack
17109
17110                                   ; ============================================================================
```

```
17111                                   ;
17112                                   ; readWriteTrack:
17113                                   ;
17114                                   ; input:
17115                                   ;    ES:di points to bds for drive
17116                                   ;    rFlag - 2 for read,3 for write
17117                                   ;
17118                                   ; ============================================================================
17119
17120                                   ReadWriteTrack:
17121                                           ; save bds pointer segment so we can use it to access
17122                                           ; our packet. Notice that this is not the standard register
17123                                           ; assignment for accessing packets
17124
17125                                           ; 19/10/2022
17126 0000119F 06                              push    es
17127 000011A0 C41E[1200]                      les     bx, [ptrsav]   ; ES:BX -> to request header
17128 000011A4 26C45F13                        les     bx, [es:bx+19] ; [es:bx+IOCTL_REQ.GENERICIOCTL_PACKET]
17129 000011A8 268B4703                        mov     ax, [es:bx+3]  ; [es:bx+A_TRACKREADWRITEPACKET.TRWP_CYLINDER]
17130 000011AC A3[3301]                        mov     [curtrk], ax
17131 000011AF 268B4701                        mov     ax, [es:bx+1]  ; [es:bx+A_TRACKREADWRITEPACKET.TRWP_HEAD]
17132 000011B3 A2[3201]                        mov     [curhd], al    ; Assume heads < 256!!!
17133 000011B6 268B4705                        mov     ax, [es:bx+5]  ; [es:bx+A_TRACKREADWRITEPACKET.TRWP_FIRSTSECTOR]
17134 000011BA 268B4F07                        mov     cx, [es:bx+7]  ; [es:bx+A_TRACKREADWRITEPACKET.TRWP_SECTORSTOREADWRITE]
17135 000011BE 26C45F09                        les     bx, [es:bx+9]  ; [es:bx+A_TRACKREADWRITEPACKET.TRWP_TRANSFERADDRESS]
17136                                                                 ; Get transfer address
17137
17138                                           ; we just trashed our packet address, but we no longer care
17139
17140 000011C2 8C06[A804]                      mov     [xfer_seg], es ; Pass transfer       segment
17141 000011C6 07                              pop     es
17142
17143                                           ; Fall into TrackIo
17144
17145                                   ; =============== S U B   R O U T I N E =======================================
17146
17147                                   ; 16/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
17148
17149                                   ; ============================================================================
17150                                   ;
17151                                   ; TrackIo:
17152                                   ;   performs track read/write/verify
17153                                   ;
17154                                   ;  input:
17155                                   ;     rFlag       - 2 = read
17156                                   ;                   3 = write
17157                                   ;                   4 = verify
17158                                   ;     AX   - Index into track table of first sector to io
17159                                   ;     CX   - Number of sectors to io
17160                                   ;     Xfer_Seg:BX - Transfer address
17161                                   ;     ES:DI       - Pointer to bds
17162                                   ;     CurTrk      - Current cylinder
17163                                   ;     CurHd       - Current head
17164                                   ;
17165                                   ; ============================================================================
17166
17167                                   ; 16/03/2019 - Retro DOS v4.0
17168
17169                                           ; 24/12/2023 - Retro DOS 5.0
17170
17171                                           ; 19/10/2022
17172                                   TrackIo:
17173                                                                   ; Procedure `disk' will pop stack to
17174 000011C7 8926[3501]                      mov     [spsav], sp    ; SpSav and return if error
17175 000011CB E8A9F7                          call    checksingle    ; Ensure correct disk is in drv
17176 000011CE 803E[A905]01                    cmp     byte [media_set_for_format], 1
17177                                                                   ; See if we have already set  disk
17178 000011D3 7407                            jz      short Dptalreadyset ; base table
17179 000011D5 50                              push    ax             ; set up tables       and variables for i/o
17180 000011D6 51                              push    cx
17181 000011D7 E8A0F9                          call    iosetup
17182 000011DA 59                              pop     cx
17183 000011DB 58                              pop     ax
17184                                   Dptalreadyset:                          ; Point si at the table entry of the
17185 000011DC BE[AC04]                        mov     si, tracktable ; first sector to be io'd
17186                                           ; 24/12/2023
17187                                           ;add    ax, ax         ; PCDOS 7.1 IBMBIO.COM
17188                                           ;add    ax, ax
17189 000011DF D1E0                            shl     ax, 1
17190 000011E1 D1E0                            shl     ax, 1
17191 000011E3 01C6                            add     si, ax
17192
17193                                           ; WE WANT:
17194                                           ; CX to be the number of times we have to loop
17195                                           ; DX to be the number of sectors we read on each iteration
17196
17197 000011E5 BA0100                          mov     dx, 1
17198
17199                                           ; 24/12/2023
17200 000011E8 26F6453F08                      test    byte [es:di+3Fh], 8 ; PCDOS 7.1 IBMBIO.COM
17201                                           ; 12/12/2022
17202                                           ;test   byte [es:di+23h], 8
17203                                           ;;test  word [es:di+35], 8 ; [es:di+BDS.flags]
17204                                                                   ; good_tracklayout
17205 000011ED 7402                            jz      short ionextsector
17206
17207 000011EF 87CA                            xchg    dx, cx         ; HEY! We can read all secs in one blow
17208                                   ionextsector:
17209 000011F1 51                              push    cx
17210 000011F2 52                              push    dx
17211 000011F3 46                              inc     si
17212 000011F4 46                              inc     si             ; Skip over the       cylinder and head in
17213                                                                   ; the track table
17214 000011F5 AC                              lodsb                  ; Get sector ID       from track table
17215 000011F6 A2[3101]                        mov     [cursec], al
17216
17217                                           ; assumptions for a fixed disk multi-track disk     i/o
17218                                           ; 1). In the input CX (# of sectors to go) to TrackIo,
17219                                           ;     only CL is valid.
17220                                           ; 2). Sector size should be set      to 512 bytes.
17221                                           ; 3). Good track layout
17222
17223                                           ; 24/12/2023
17224 000011F9 26F6453F01                      test    byte [es:di+3Fh], 1 ; PCDOS 7.1 IBMBIO.COM
17225                                           ; 12/12/2022
17226                                           ;test   byte [es:di+23h], 1
17227                                           ;;test  word [es:di+35], 1 ; [es:di+BDS.flags]
17228                                                                   ; fnon_removable ; Fixed disk?
17229 000011FE 7414                            jz      short IoRemovable ; No
17230
17231                                           ; 12/12/2022
17232 00001200 F606[A004]80                    test    byte [multrk_flag], 80h
17233                                           ;test   word [multrk_flag], 80h ; MULTI_TRK_ON
17234                                                                     ; Allow multi-track operation?
```

```
17235 00001205 740D                           jz      short IoRemovable ; No,don't do that.
17236 00001207 8916[2201]                     mov     [seccnt], dx
17237 0000120B 89D0                           mov     ax, dx
17238 0000120D E823FA                         call    Disk
17239 00001210 5A                             pop     dx
17240 00001211 59                             pop     cx
17241 00001212 F8                             clc
17242 00001213 C3                             retn
17243                                 ; ---------------------------------------------------------------------------
17244
17245                                 IoRemovable:
17246 00001214 AC                             lodsb                   ; Get sector size index       from track
17247                                                                   ; table and save it
17248 00001215 50                             push    ax
17249 00001216 56                             push    si
17250 00001217 1E                             push    ds              ; Save Bios_Data
17251 00001218 50                             push    ax
17252 00001219 8A26[2C01]                     mov     ah, [eot]       ; Preserve whatever might be in      ah
17253                                                                   ; Fetch EOT while ds-> Bios_Data
17254 0000121D C536[2D01]                     lds     si, [dpt]
17255 00001221 884403                         mov     [si+3], al      ; [si+DISK_PARMS.DISK_SECTOR_SIZ]
17256 00001224 886404                         mov     [si+4], ah      ; [si+DISK_PARMS.DISK_EOT]
17257 00001227 58                             pop     ax
17258 00001228 1F                             pop     ds
17259 00001229 88D0                           mov     al, dl
17260 0000122B A3[2201]                       mov     [seccnt], ax
17261 0000122E E802FA                         call    Disk
17262 00001231 5E                             pop     si              ; Advance buffer pointer by adding
17263                                                                   ; sector size
17264                                         ;pop    ax
17265                                         ; 24/12/2023
17266 00001232 59                             pop     cx
17267
17268                                         ; 02/09/2023 (PCDOS 7.1)
17269                                         ;call   SectorSizeIndexToSectorSize
17270                                         ;mov    cl, al  ; 24/12/2023
17271 00001233 B88000                         mov     ax, 128
17272 00001236 D3E0                           shl     ax, cl
17273
17274 00001238 01C3                           add     bx, ax
17275 0000123A 5A                             pop     dx
17276 0000123B 59                             pop     cx
17277 0000123C E2B3                           loop    ionextsector
17278 0000123E 803E[A905]01                   cmp     byte [media_set_for_format], 1
17279                                         ;jz     short NoNeedDone
17280                                         ; 12/12/2022
17281 00001243 7404                           je      short NoNeedDone2
17282 00001245 E877F9                         call    done            ; set time of last access, and reset
17283                                                                   ; entries in Dpt.
17284                                 NoNeedDone:
17285 00001248 F8                             clc     ; not necessary ('done' clears cf) ; 24/12/2023
17286                                 NoNeedDone2:
17287 00001249 C3                             retn
17288
17289                                 ; =============== S U B   R O U T I N E =======================================
17290
17291                                 ; ---------------------------------------------------------------------------
17292                                 ;
17293                                 ; The sector size in bytes needs to be converted to an index value for the ibm
17294                                 ; rom. (0=>128,1=>256,2=>512,3=>1024). It is assumed that only these values
17295                                 ; are permissible.
17296                                 ;
17297                                 ; On Input   AX contains sector size in bytes
17298                                 ; On Output  AL Contains index
17299                                 ; All other registers preserved
17300                                 ;
17301                                 ; ---------------------------------------------------------------------------
17302
17303                                 ; 02/09/2023
17304                                 ;SectSizeToSectIndex:
17305                                 ;           cmp     ah, 2           ; (0=>128,1=>256,2=>512,3=>1024)
17306                                 ;                                   ; examine upper       byte only
17307                                 ;           ja      short OneK
17308                                 ;           mov     al, ah          ; value in AH is the index!
17309                                 ;           retn
17310
17311                                 ; ---------------------------------------------------------------------------
17312                                 ;
17313                                 ;OneK:
17314                                 ;           mov     al, 3
17315                                 ;           retn
17316
17317                                 ; =============== S U B   R O U T I N E =======================================
17318
17319                                 ; 02/09/2023
17320                                 ;SectorSizeIndexToSectorSize:
17321                                 ;           mov     cl, al
17322                                 ;           mov     ax, 128
17323                                 ;           shl     ax, cl
17324                                 ;           retn
17325
17326                                 ; =============== S U B   R O U T I N E =======================================
17327
17328                                 ; 16/10/2022
17329
17330                                 ; ---------------------------------------------------------------------------
17331                                 ;
17332                                 ; SetDASD
17333                                 ;
17334                                 ; Set up the rom for formatting.
17335                                 ; we have to tell the rom bios what type of disk is in the drive.
17336                                 ;
17337                                 ; On Input   - ES:di - Points to bds
17338                                 ;
17339                                 ; ---------------------------------------------------------------------------
17340
17341                                           ; 24/12/2023 - Retro DOS 5.0
17342                                           ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:129Bh)
17343
17344                                           ; 19/10/2022
17345                                 SetDasd:
17346 0000124A 803E[AA05]01                     cmp     byte [had_format_error], 1 ;
17347                                                                   ; See if we've previously set dasd type
17348 0000124F 740C                             jz      short DoSetDasd
17349                                           ; 24/12/2023
17350 00001251 26F6453F80                       test    byte [es:di+3Fh], 80h
17351                                           ; 10/12/2022
17352                                           ;test   byte [es:di+23h], 80h
17353                                           ;;test  word [es:di+23h], 80h ; [es:di+BDS.flags]
17354                                                                   ; set_dasd_true
17355 00001256 7446                             jz      short DasdHasBeenSet
17356                                           ; 24/12/2023
17357 00001258 2680653F7F                       and     byte [es:di+3Fh], 7Fh
17358                                           ; 10/12/2022
```

```
17359                                           ;and   byte [es:di+23h], 7Fh
17360                                           ;;and  word [es:di+23h], 0FF7Fh ; [es:di+BDS.flags]
17361                                                                   ; ~set_dasd_true
17362                                   DoSetDasd:
17363 0000125D C606[AA05]00                       mov    byte [had_format_error], 0 ; Reset it
17364 00001262 C606[3B01]50                       mov    byte [gap_patch], 50h ; Format gap for 48tpi disks
17365 00001267 B004                               mov    al, 4
17366                                             ; 24/12/2023
17367 00001269 268A653E                           mov    ah, [es:di+3Eh]
17368                                             ; 02/09/2023
17369                                             ;mov   ah, [es:di+22h] ; [es:di+BDS.formfactor]
17370 0000126D 80FC02                             cmp    ah, 2
17371                                             ;cmp   byte [es:di+22h], 2 ; [es:di+BDS.formfactor]
17372                                                                   ; DEV_3INCH720KB
17373 00001270 7414                               jz     short DoSet
17374                                             ; 24/12/2023
17375 00001272 B001                               mov    al, 1
17376                                             ;cmp   ah, 1
17377 00001274 38C4                               cmp    ah, al  ; 1
17378                                             ;cmp   byte [es:di+22h], 1 ; [es:di+BDS.formfactor]
17379                                                                   ; DEV_5INCH96TPI
17380                                             ;jz    short GotBig
17381                                             ; 24/12/2023
17382                                             ;mov   al, 1
17383                                             ;jmp   short DoSet
17384                                             ; 02/09/2023
17385 00001276 750E                               jnz    short DoSet
17386                                   GotBig:
17387                                             ;mov   al, 2          ; 160/320k in a       1.2 meg drive
17388                                             ; 02/09/2023
17389 00001278 40                                 inc    ax  ; mov al, 2
17390 00001279 803E[A805]00                       cmp    byte [mediatype], 0
17391 0000127E 7506                               jnz    short DoSet
17392                                             ;mov   al, 3          ; 1.2meg in a 1.2meg drive
17393                                             ; 10/12/2022
17394                                             ;inc   al  ; al = 3
17395                                             ; 18/12/2022
17396 00001280 40                                 inc    ax  ; al = 3
17397 00001281 C606[3B01]54                       mov    byte [gap_patch], 54h
17398                                   DoSet:
17399 00001286 1E                                 push   ds
17400 00001287 56                                 push   si
17401
17402                                             ;mov   ds, [zeroseg]  ; Point to interrupt vectors
17403                                             ; 02/09/2023
17404 00001288 31F6                               xor    si, si
17405 0000128A 8EDE                               mov    ds, si  ; 0
17406
17407 0000128C C5367800                           lds    si, [DSKADR]
17408                                             ;lds   si, [78h]      ; [DSKADR]  (Int 1Eh)
17409                                             ;;lds  si, ds:78h
17410
17411 00001290 C644090F                           mov    byte [si+9], 0Fh ;
17412                                                                   ; [si+DISK_PARMS.DISK_HEAD_STTL]
17413 00001294 5E                                 pop    si
17414 00001295 1F                                 pop    ds
17415 00001296 B417                               mov    ah, 17h
17416 00001298 268A5504                           mov    dl, [es:di+4]
17417 0000129C CD13                               int    13h            ; DISK - DISK -       SET TYPE (AT,XT2,XT286,CONV,PS
17418                                                                   ; AL = disk type AL = 03h - high-capacity disk in high-capacity   drive
17419                                   DasdHasBeenSet:
17420 0000129E 268A6513                           mov    ah, [es:di+13h]       ; [es:di+BDS.secpertrack]
17421 000012A2 8826[3701]                         mov    [formt_eot], ah
17422 000012A6 C3                                 retn
17423
17424                                   ; ============== S U B   R O U T I N E =======================================
17425
17426                                   ; 16/10/2022
17427
17428                                   ; ---------------------------------------------------------------------------
17429                                   ;
17430                                   ; Set Media Type for Format
17431                                   ; Performs the int 13 with ah = 18h to see if the medium described in the
17432                                   ; BPB area in the BDS can be handled by the rom.
17433                                   ; On Input, ES:DI -> current BDS.
17434                                   ; The status of the operation is returned in AL
17435                                   ;
17436                                   ;   - 0 - if the support is available,and the combination is valid.
17437                                   ;   - 1 - no rom support
17438                                   ;   - 2 - illegal combination
17439                                   ;   - 3 - no media present (rom support exists but cannot determine now)
17440                                   ;
17441                                   ; Flags also may be altered. All other registers preserved.
17442                                   ; If the call to rom returns no error,then the current Dpt is "replaced" by
17443                                   ; the one returned by the rom. This is Done by changing the pointer in [Dpt]
17444                                   ; to the one returned. the original pointer to the disk base table is stored
17445                                   ; in TempDpt, until it is restored.
17446                                   ;
17447                                   ; ---------------------------------------------------------------------------
17448
17449                                             ; 24/12/2023 - Retro DOS 5.0
17450
17451                                             ; 19/10/2022
17452                                   SetMediaForFormat:
17453 000012A7 51                                 push   cx
17454 000012A8 52                                 push   dx
17455
17456                                             ; If we have a format error, then do not change Dpt, TempDpt.
17457                                             ; but we need to call int 13h, ah=18h again.
17458
17459 000012A9 803E[AA05]01                       cmp    byte [had_format_error], 1
17460 000012AE 7425                               jz     short SkipSaveDskAdr
17461 000012B0 30C0                               xor    al, al         ; If already done return 0
17462 000012B2 803E[A905]01                       cmp    byte [media_set_for_format], 1
17463 000012B7 7502                               jnz    short DoSetMediaForFormat
17464 000012B9 EB7D                               jmp    SetMediaRet    ; Media already set
17465                                   ; ---------------------------------------------------------------------------
17466
17467                                   DoSetMediaForFormat:
17468 000012BB 06                                 push   es
17469 000012BC 56                                 push   si
17470
17471                                             ; 02/09/2023
17472                                             ;mov   es, [zeroseg]  ; Point to interrupt vectors
17473 000012BD 31F6                               xor    si, si ; 0
17474 000012BF 8EC6                               mov    es, si
17475
17476 000012C1 26C4367800                         les    si, [es:DSKADR]
17477                                             ;les   si, es:78h     ; [es:DSKADR]
17478                                                                   ; Get pointer to disk base table
17479 000012C6 8936[2D01]                         mov    [dpt], si
17480 000012CA 8C06[2F01]                         mov    [dpt+2], es    ; Save pointer to table
17481
17482                                             ; Initialize the head settle time to 0Fh. See the offsets
```

```
                                        ; given in dskprm.inc.
                                        
000012CE 26C644090F                     mov     byte [es:si+9], 0Fh ; [es:si+DISK_PARMS.DISK_HEAD_STTL]
000012D3 5E                             pop     si
000012D4 07                             pop     es
                                SkipSaveDskAdr:
                                        ; 24/12/2023
000012D5 268B4D41                       mov     cx, [es:di+41h]         ; (PCDOS 7.1 IBMBIO.COM)
                                        ;mov    cx, [es:di+25h]         ; [es:di+BDS.cylinders]
000012D9 49                             dec     cx
000012DA 80E503                         and     ch, 3
000012DD D0CD                           ror     ch, 1
000012DF D0CD                           ror     ch, 1
000012E1 86CD                           xchg    ch, cl
000012E3 260A4D13                       or      cl, [es:di+13h]         ; [es:di+BDS.secpertrack]
000012E7 268A5504                       mov     dl, [es:di+4]  ; [es:di+BDS.drivenum]
000012EB 06                             push    es
000012EC 1E                             push    ds
000012ED 56                             push    si
000012EE 57                             push    di
000012EF B418                           mov     ah, 18h
000012F1 CD13                           int     13h             ; DISK - SET MEDIA TYPE       FOR FORMAT (AT model 3x9,XT2,XT286,PS)
                                                                ; DL = drive number, CH       = lower 8 bits of number of tracks, CL = sectors
per track
000012F3 7231                           jc      short FormaStatErr
000012F5 803E[AA05]01                   cmp     byte [had_format_error], 1
000012FA 7423                           jz      short skip_disk_base_setting
000012FC 06                             push    es              ; Save segment returned       by the rom
                                        
                                        ; 02/09/2023
                                        ;mov    es, [zeroseg]  ; Point to interrupt vector segment
00001301 31F6                           xor     si, si
00001303 8EC6                           mov     es, si ; 0
00001305 06                             push    es ; * ; 02/09/2023
                                        
00001306 26C4367800                     les     si, [es:DSKADR]
                                        ;les    si, es:78h      ; [es:DSKADR] (Int 1Eh)
                                                                ; Get current disk base       table
0000130B 8936[AB05]                     mov     [tempdpt], si
0000130F 8C06[AD05]                     mov     [tempdpt+2], es ; Save it
                                        
                                        ; 02/09/2023
                                        ;;mov   es, [zeroseg]
                                        ;xor    si, si ; 0
                                        ;mov    es, si
00001313 07                             pop     es ; * ; 02/09/2023
                                        
                                        ;mov    es:78h, di
00001314 26893E7800                     mov     [es:DSKADR], di
                                        ;pop    word ptr es:7Ah         ; replace with one returned by rom
00001319 268F067A00                     pop     word [es:DSKADR+2]
0000131E C606[A905]01                   mov     byte [media_set_for_format], 1
                                skip_disk_base_setting:
00001323 30C0                           xor     al, al          ; Legal combination + rom support code
                                        ;mov    ds:had_format_error, al       ; Reset the flag
00001325 A2[AA05]                       mov     [had_format_error], al
00001328 EB0E                           jmp     short PopStatRet
                                ; ---------------------------------------------------------------------------
                                
                                FormaStatErr:
                                        ; 10/12/2022
0000132A B003                           mov     al, 3
                                        
0000132C 80FC0C                         cmp     ah, 0Ch         ; DSK_ILLEGAL_COMBINATION
                                                                ; Illegal combination =       0Ch
0000132F 7406                           jz      short FormatStatIllegalComb
00001331 80FC80                         cmp     ah, 80h         ; DSK_TIMEOUT_ERR
00001334 7402                           jz      short FormatStatTimeOut
                                        ; 10/12/2022
                                        ;dec    al
                                        ; 18/12/2022
00001336 48                             dec     ax
                                        ; al = 2
                                        ;mov    al, 1           ; Function not supported.
                                        ;jmp    short PopStatRet
                                ; ---------------------------------------------------------------------------
                                
                                FormatStatIllegalComb:
                                        ; 10/12/2022
                                        ;dec    al      ; 3 -> 2 or 2 -> 1
                                        ; 18/12/2022
00001337 48                             dec     ax
                                        ; al = 2
                                        ;mov    al, 2           ; Function supported, but
                                                                ; Illegal sect/trk,trk combination.
                                        ; 10/12/2022
                                        ;jmp    short PopStatRet
                                ; ---------------------------------------------------------------------------
                                
                                FormatStatTimeOut:
                                        ; 10/12/2022
                                        ; al = 3
                                        ;mov    al, 3           ; Function supported, but
                                                                ; Media not present.
                                PopStatRet:
00001338 5F                             pop     di
00001339 5E                             pop     si
0000133A 1F                             pop     ds
0000133B 07                             pop     es
                                SetMediaRet:
0000133C 5A                             pop     dx
0000133D 59                             pop     cx
0000133E C3                             retn
                                
                                ; =============== S U B   R O U T I N E =======================================
                                
                                ; 16/10/2022
                                
                                ; ---------------------------------------------------------------------------
                                ;
                                ; RESET THE DRIVE
                                ;
                                ; we also set [Step_Drv] to -1 to force the main disk routine to use the
                                ; slow head settle time for the next operation. this is because the reset
                                ; operation moves the head to cylinder 0,so we need to do a seek the next
                                ; time around - there is a problem with 3.5" drives in that the head does
                                ; not settle down in time,even for read operations!!
                                ;
                                ; ---------------------------------------------------------------------------
                                
                                ResetDisk:
0000133F 50                             push    ax
                                        
                                        ; 02/09/2023
```

```
17606 0000133C B80100                           mov     ax, 1 ; PCDOS 7.1
17607 0000133F 3806[A905]                       cmp     [media_set_for_format], al ; 1
17608                                           ;cmp    byte [media_set_for_format], 1
17609                                                                   ; Reset while formatting?
17610 00001343 7503                             jnz     short ResetDisk_cont
17611                                                                   ; Then verify operation in "fmt & vrfy"
17612                                           ;mov    byte [had_format_error], 1 ; Might have failed.
17613 00001345 A2[AA05]                         mov     [had_format_error], al ; 1
17614                                   ResetDisk_cont:
17615                                           ; 02/09/2023 (ah=0)
17616                                           ;xor    ah, ah          ; So signals that we had a format error
17617 00001348 CD13                             int     13h             ; DISK - RESET DISK SYSTEM
17618                                                                   ; DL = drive (if bit 7 is set both hard     disks and floppy disks reset)
17619 0000134A C606[7600]FF                     mov     byte [step_drv], 0FFh ; -1
17620                                                                   ; Zap up the speed
17621 0000134F 58                               pop     ax
17622 00001350 C3                               retn
17623
17624                                   ; =============== S U B   R O U T I N E =======================================
17625
17626                                   ; 16/10/2022
17627
17628                                   ; ---------------------------------------------------------------------------
17629                                   ;
17630                                   ; This routine sets up the drive parameter table with the values needed for
17631                                   ; format,does an int 13. values in Dpt are restored after a verify is done.
17632                                   ;
17633                                   ; on entry  -      ES:DI - points to bds for the drive
17634                                   ;           Xfer_Seg:BX - points to trkbuf
17635                                   ;           AL     - number of sectors
17636                                   ;           AH     - int 13 function code
17637                                   ;           CL     - sector number for verify
17638                                   ;           DS     - Bios_Data
17639                                   ;
17640                                   ; ON EXIT   -      DS,DI,ES,BX remain unchanged.
17641                                   ;           AX and flags are the results of the int 13
17642                                   ;
17643                                   ; ---------------------------------------------------------------------------
17644
17645                                           ; 24/12/2023 - Retro DOS 5.0
17646
17647                                           ; 19/10/2022
17648                                   ToRom:
17649 00001351 53                               push    bx
17650 00001352 56                               push    si
17651
17652                                           ; Compaq bug fix - check whether we are using new ROM
17653                                           ; functionality to set up format, not merely if it exists.
17654                                           ; This was formerly a check against [new_rom]
17655
17656 00001353 F606[A905]01                      test    byte [media_set_for_format], 1
17657 00001358 7534                             jnz     short GotValidDpt
17658 0000135A 50                               push    ax
17659 0000135B 06                               push    es              ; Save bds segment
17660                                           ; 24/12/2023
17661 0000135C 26807D3E02                       cmp     byte [es:di+3Eh], 2
17662                                           ;cmp    byte [es:di+22h], 2 ; [es:di+BDS.formfactor]
17663                                                                   ; ffSmall ; is it a 3.5" drive?
17664                                           ; 24/12/2023
17665                                           ;pushf  ; not necessary         ; (Save the cmp result)
17666 00001361 8E06[1A00]                       mov     es, [zeroseg]
17667                                           ;les    si, es:78h      ; Get pointer to disk base table
17668 00001365 26C4367800                       les     si, [es:DSKADR]
17669                                           ;mov    word ptr ds:dpt, si
17670                                           ;mov    word ptr ds:dpt+2, es ;         Save pointer to table
17671 0000136A 8936[2D01]                       mov     [dpt], si
17672 0000136E 8C06[2F01]                       mov     [dpt+2], es     ; Save pointer to table
17673
17674 00001372 A0[3701]                         mov     al, [formt_eot]
17675 00001375 26884404                         mov     [es:si+4], al   ; [es:si+DISK_PARMS.DISK_EOT]
17676 00001379 A0[3B01]                         mov     al, [gap_patch]
17677 0000137C 26884407                         mov     [es:si+7], al   ; [es:si+DISK_PARMS.DISK_FORMT_GAP]
17678                                                                   ; Important for         format
17679 00001380 26C644090F                       mov     byte [es:si+9], 0Fh ; [es:si+DISK_PARMS.DISK_HEAD_STTL]
17680                                                                   ; Assume we are         doing a seek operation
17681                                                                   ; Setup motor start correctly for 3.5" drives
17682                                           ; 24/12/2023
17683                                           ;popf                   ; Get result of         earlier cmp
17684 00001385 7505                             jnz     short MotorStrtOK
17685 00001387 26C6440A04                       mov     byte [es:si+0Ah], 4 ; [es:si+DISK_PARMS.DISK_MOTOR_STRT]
17686                                   MotorStrtOK:
17687 0000138C 07                               pop     es              ; Restore bds segment
17688 0000138D 58                               pop     ax
17689                                   GotValidDpt:
17690 0000138E 8B16[3901]                       mov     dx, [trknum]    ; Set track number
17691 00001392 88D5                             mov     ch, dl          ; Set low 8 bits in ch
17692 00001394 268A5504                         mov     dl, [es:di+4]   ; Set drive number
17693 00001398 8A36[3801]                       mov     dh, [hdnum]     ; Set head number
17694 0000139C 06                               push    es              ; Save bds segment
17695 0000139D 8E06[A804]                       mov     es, [xfer_seg]
17696 000013A1 CD13                             int     13h             ; DISK -
17697 000013A3 07                               pop     es              ; Restore bds segment
17698 000013A4 5E                               pop     si
17699 000013A5 5B                               pop     bx
17700 000013A6 C3                               retn
17701
17702                                   ; ---------------------------------------------------------------------------
17703
17704                                   ; 16/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
17705                                   ; 24/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
17706
17707                                   ; BIOSCODE:1124h (MSDOS 6.21, IO.SYS)
17708                                   ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:1404h
17709
17710                                   ; ===========================================================================
17711                                   ;
17712                                   ; get the owner of the physical drive represented by the logical drive in al.
17713                                   ; the assumption is that we **always** keep track of the owner of a drive!!
17714                                   ; if this is not the case, the system may hang, just following the linked list.
17715                                   ;
17716                                   ; ===========================================================================
17717
17718                                           ; 24/12/2023 - Retro DOS 5.0
17719
17720                                           ; 19/10/2022
17721                                   ioctl_getown:
17722 000013A7 E8FAF1                           call    SetDrive
17723 000013AA 268A4504                         mov     al, [es:di+4]   ; [es:di+BDS.drivenum]
17724                                                                   ; Get physical drive number
17725 000013AE C43E[1901]                       les     di, [start_bds] ; Get start of bds chain
17726                                   ownloop:
17727 000013B2 26384504                         cmp     [es:di+4], al   ; [es:di+BDS.drivenum]
17728 000013B6 7507                             jnz     short getnextBDS
17729                                           ; 24/12/2023
```

```
17730 000013B8 26F6453F20                       test    byte [es:di+3Fh], 20h ; (PCDOS 7.1 IBMBIO.COM)
17731                                           ; 10/12/2022
17732                                           ;test  byte [es:di+23h], 20h
17733                                           ;;test  word [es:di+23h], 20h ; [es:di+BDS.flags]
17734                                                   ; fi_own_physical
17735 000013BD 7514                             jnz     short exitown
17736                                   getnextBDS:
17737 000013BF 26C43D                           les     di, [es:di]     ; [es:di+BDS.link]
17738 000013C2 EBEE                             jmp     short ownloop
17739                                   ; ---------------------------------------------------------------------------
17740
17741                                   ; =============================================================================
17742                                   ;
17743                                   ; set the ownership of the physical drive represented by the logical drive
17744                                   ; in al to al.
17745                                   ;
17746                                   ; =============================================================================
17747
17748                                           ; 24/12/2023 - Retro DOS 5.0
17749
17750                                           ; 19/10/2022
17751                                   ioctl_setown:
17752 000013C4 E8DDF1                           call    SetDrive
17753 000013C7 C606[7A00]01                     mov     byte [fsetowner], 1
17754                                                   ; set flag for CheckSingle to look at.
17755 000013CC E8A8F5                           call    checksingle
17756                                           ; 02/09/2023
17757 000013CF FE0E[7A00]                       dec     byte [fsetowner] ; 0
17758                                           ;mov    byte [fsetowner], 0
17759                                                   ; set ownership       of drive reset flag
17760                                           ; Fall into ExitOwn
17761
17762                                   ; =============================================================================
17763                                   ;
17764                                   ; if there is only one logical drive assigned to this physical drive, return
17765                                   ; 0 to user to indicate this. Enter with ES:di -> the owner's bds.
17766                                   ;
17767                                   ; =============================================================================
17768
17769                                           ; 24/12/2023 - Retro DOS 5.0
17770                                   exitown:
17771 000013D3 30C9                             xor     cl, cl
17772                                           ; 24/12/2023
17773 000013D5 26F6453F10                       test    byte [es:di+3Fh], 10h ; (PCDOS 7.1 IBMBIO.COM)
17774                                           ; 12/12/2022
17775                                           ;test  byte [es:di+23h], 10h
17776                                           ;;test  word [es:di+23h], 10h ; [es:di+BDS.flags]
17777                                                   ; fi_am_mult
17778 000013DA 7406                             jz      short exitnomult
17779 000013DC 268A4D05                         mov     cl, [es:di+5]  ; [es:di+BDS.drivelet]
17780                                                   ; Get logical drive number
17781                                                   ; Get it 1-based
17782 000013E0 FEC1                             inc     cl
17783                                   exitnomult:
17784 000013E2 C51E[1200]                       lds     bx, [ptrsav]
17785 000013E6 884F01                           mov     [bx+1],cl       ; [bx+unit]
17786                                                   ; Exit normal termination
17787                                           ; 12/12/2022
17788                                           ; cf=0
17789                                           ;clc
17790 000013E9 C3                               retn
17791
17792                                   ; =============== S U B   R O U T I N E =======================================
17793
17794                                   ; 16/10/2022
17795
17796                                   ; ---------------------------------------------------------------------------
17797                                   ;
17798                                   ; moves the old Dpt that had been saved in TempDpt back to Dpt. this is done
17799                                   ; only if the first byte of TempDpt is not -1.
17800                                   ; all registers (including flags) are preserved.
17801                                   ;
17802                                   ; ---------------------------------------------------------------------------
17803
17804                                           ; 24/12/2023
17805                                           ; 19/10/2022
17806                                   RestoreOldDpt:
17807                                           ; if we have already restored the disk base table earlier,
17808                                           ; do not do it again.
17809
17810 000013EA 50                               push    ax
17811 000013EB 30C0                             xor     al, al
17812 000013ED A2[AA05]                         mov     [had_format_error], al; Reset flag and
17813 000013F0 8606[A905]                       xchg    al, [media_set_for_format] ; get current flag setting
17814 000013F4 08C0                             or      al, al
17815 000013F6 7418                             jz      short DontRestore
17816 000013F8 56                               push    si
17817 000013F9 1E                               push    ds
17818 000013FA 06                               push    es
17819 000013FB C536[AB05]                       lds     si, [tempdpt]
17820
17821                                           ; 17/10/2022
17822                                           ;mov    es, [cs:BIOSDATAWORD]
17823                                           ;;mov   es, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
17824                                           ;mov    es, [es:zeroseg]
17825                                           ;;mov   es, es:zeroseg ; CAS -- bleeeech!
17826
17827                                           ; 24/12/2023
17828 000013FF 31C0                             xor     ax, ax
17829 00001401 8EC0                             mov     es, ax ; 0
17830
17831                                           ;mov    es:78h,si       ; [es:DSKADR]  (Int 1Eh)
17832 00001403 2689367800                       mov     [es:DSKADR], si
17833                                           ;mov    word ptr es:7Ah, ds ; [es:DSKADR+2]
17834 00001408 268C1E7A00                       mov     [es:DSKADR+2], ds
17835 0000140D 07                               pop     es
17836 0000140E 1F                               pop     ds
17837 0000140F 5E                               pop     si
17838                                   DontRestore:
17839 00001410 58                               pop     ax
17840                                           ; 12/12/2022
17841                                           ; cf=0
17842                                           ;clc                    ;  Clear carry
17843 00001411 C3                               retn
17844
17845                                   ; ---------------------------------------------------------------------------
17846
17847                                   ; 16/10/2022
17848
17849                                   ; =============================================================================
17850                                   ;   get media id
17851                                   ; =============================================================================
17852                                   ;
17853                                   ; FUNCTION: get the volume label,the system id and the serial number from
```

```
17854                                   ;       the media that has the extended boot record.
17855                                   ;       for the conventional media,this routine will return "unknown
17856                                   ;       media type" error to dos.
17857                                   ;
17858                                   ; INPUT :   ES:di -> bds table for this drive.
17859                                   ;
17860                                   ; OUTPUT:   the request packet filled with the information,if not carry.
17861                                   ;       if carry set,then al contains the device driver error number
17862                                   ;       that will be returned to dos.
17863                                   ;       register DS,DX,AX,CX,DI,SI destroyed.
17864                                   ;
17865                                   ; SUBROUTINES TO BE CALLED:
17866                                   ;   BootIo:NEAR
17867                                   ;
17868                                   ; LOGIC:
17869                                   ;   to recognize the extended boot record,this logic will actually
17870                                   ;   access the boot sector even if it is a hard disk.
17871                                   ;   note:the valid extended bpb is recognized by looking at the mediabyte
17872                                   ;   field of bpb and the extended boot signature.
17873                                   ;
17874                                   ; {
17875                                   ;   get logical drive number from bds table;
17876                                   ;   rFlag = read operation;
17877                                   ;   BootIo;         /*get the media boot record into the buffer
17878                                   ;   if (no error) then
17879                                   ;         if (extended boot record) then
17880                                   ;           { set volume label,volume serial number and system id
17881                                   ;             of the request packet to those of the boot record;
17882                                   ;           };
17883                                   ;         else               /*not an extended bpb */
17884                                   ;           { set register al to "unknown media.." error code;
17885                                   ;             set carry bit;
17886                                   ;           };
17887                                   ;   else
17888                                   ;         ret;      /*already error code is set in the register al
17889                                   ;
17890                                   ; ============================================================================
17891
17892                                   ;size_of_EXT_BOOT_SERIAL equ 4
17893                                   ;;size_of_EXT_BOOT_VOL_LABEL equ 11
17894                                   ;;size_of_EXT_SYSTEM_ID equ 8
17895
17896                                           ; 24/12/2023 - Retro DOS 5.0
17897                                           ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:1478h)
17898
17899                                           ; 19/10/2022
17900                                   GetMediaId:
17901 00001412 E8B000                             call    ChangeLineChk
17902 00001415 268A4505                           mov     al, [es:di+5]  ; [es:di+BDS.drivelet] ; Logical drive number
17903 00001419 C606[2001]02                       mov     byte [rflag], 2       ; Read operation
17904 0000141E E88C00                             call    BootIo         ; Read boot sector into      DiskSector
17905 00001421 722E                               jb      short IOCtl_If1
17906                                                                    ; Valid? (0F0h-0FFh?)
17907 00001423 803E[6701]F0                       cmp     byte [disksector+15h], 0F0h
17908                                                                    ; [disksector+EXT_BOOT.BPB+EBPB.MEDIADESCRIPTOR]
17909                                             ;jb     short IOCtl_If2       ; brif not valid (0F0h - 0FFh)
17910                                             ; 24/12/2023
17911 00001428 7225                               jb      short IOCtl_If7
17912
17913                                           ; 24/12/2023
17914                                           ; 10/12/2022
17915                                           ;mov    si, disksector+26h
17916                                           ;;;
17917                                           ; 24/12/2023
17918                                           ;mov    si, disksector+43h ; BS_FAT32_VolID
17919 0000142A BE[9401]                           mov     si, disksector+42h ; BS_FAT32_BootSig ; 24/12/2023
17920 0000142D 833E[6801]00                       cmp     word [disksector+16h], 0 ; BPB.FATSz16
17921 00001432 7403                               jz      short IOCtl_If3 ; FAT32 fs
17922 00001434 83EE1C                             sub     si, 1Ch         ; FAT (12-16) fs ; 43h-1Ch = 27h ; BS_VolID
17923                                           ; si = disksector+26h = BS_BootSig ; 24/12/2023
17924                                   IOCtl_If3:
17925                                           ;cmp    byte [si-1], 29h ; BS_BootSig or BS_FAT32_BootSig
17926                                           ;;;
17927 00001437 803C29                             cmp     byte [si], 29h
17928                                           ;cmp    byte [disksector+26h], 29h ; [disksector+EXT_BOOT.SIG]
17929                                                                    ; EXT_BOOT_SIGNATURE
17930 0000143A 7512                               jne     short IOCtl_If2       ; not extended boot record
17931 0000143C C43E[1200]                         les     di, [ptrsav]   ; es:di points to request header
17932 00001440 26C47F13                           les     di, [es:bx+19] ; [es:bx+IOCTL_REQ.GENERICIOCTL_PACKET]
17933                                           ; 10/12/2022
17934                                           ;mov    si, disksector+27h ; disksector+EXT_BOOT.SERIAL
17935 00001444 46                                 inc     si
17936                                           ; 24/12/2023
17937                                           ; si = disksector+27h (BS_VolID)
17938                                           ;      or disksector+43h (BS_FAT32_VolID)
17939
17940 00001445 83C702                             add     di, 2          ; A_MEDIA_ID_INFO.MI_SERIAL
17941                                   IOCtl_If4:      ; 24/12/2023
17942 00001448 B91700                             mov     cx, 23         ; size_of_EXT_BOOT_SERIAL
17943                                                                    ; L+size_of_EXT_BOOT_VOL_LABEL
17944                                                                    ; +size_of_EXT_SYSTEM_ID
17945 0000144B F3A4                               rep movsb              ; Move from Bios_Data into request packet
17946
17947                                           ; 10/12/2022
17948                                           ; cf = 0
17949                                           ;clc
17950
17951 0000144D C3                                 retn
17952                                   ; ----------------------------------------------------------------------------
17953
17954                                           ; 24/12/2023
17955                                   IOCtl_If2:
17956 0000144E F9                                 stc
17957                                   IOCtl_If7:
17958 0000144F B007                               mov     al, 7          ; error_unknown_media
17959                                             ;stc
17960                                   IOCtl_If6:
17961                                   IOCtl_If1:
17962 00001451 C3                                 retn
17963                                   ; ----------------------------------------------------------------------------
17964
17965                                   ; 16/10/2022
17966
17967                                   ; ============================================================================
17968                                   ;  set media id
17969                                   ; ============================================================================
17970
17971                                   ; function: set the volume label, the system id and the serial number of
17972                                   ;       the media that has the extended boot record.
17973                                   ;       for the conventional media, this routine will return "unknown
17974                                   ;       media.." error to dos.
17975                                   ;       this routine will also set the corresponding informations in
17976                                   ;       the bds table.
17977                                   ;
```

```
                        ; input :   ES:di -> bds table for this drive.
                        ;
                        ; output:   the extended boot record in the media will be set according to
                        ;       the request packet.
                        ;       if carry set, then al contains the device driver error number
                        ;       that will be returned to dos.
                        ;
                        ; subroutines to be called:
                        ;   BootIo:NEAR
                        ;
                        ; logic:
                        ;
                        ; {
                        ;   get drive_number from bds;
                        ;   rFlag = "read operation";
                        ;   BootIo;
                        ;   if (no error) then
                        ;        if (extended boot record) then
                        ;          { set volume label,volume serial number and system id
                        ;            of the boot record to those of the request packet;
                        ;            rFlag = "write operation";
                        ;            get drive number from bds;
                        ;            BootIo;        /*write it back*/
                        ;          };
                        ;        else               /*not an extended bpb */
                        ;          { set register al to "unknown media.." error code;
                        ;            set carry bit;
                        ;            ret;  /*return back to caller */
                        ;          };
                        ;   else
                        ;        ret;              /*already error code is set */
                        ;
                        ; ============================================================================

                                  ; 24/12/2023 - Retro DOS 5.0

                                  ; 19/10/2022
                        SetMediaId:
00001452 E87000                   call    ChangeLineChk
00001455 268A4505                 mov     al, [es:di+5]  ; [es:di+BDS.drivelet]
                                                         ; Logical drive       number
00001459 88C2                     mov     dl, al
0000145B C606[2001]02             mov     byte [rflag], 2       ; romread
00001460 52                       push    dx
00001461 E84900                   call    BootIo         ; Read boot sec       to Bios_Data:DiskSector
00001464 5A                       pop     dx
00001465 72EA                     jb      short IOCtl_If6
                                                         ; Valid? (0F0h-0FFh?)
00001467 803E[6701]F0             cmp     byte [disksector+15h], 0F0h
                                                         ; [disksector+EXT_BOOT.BPB+EBPB.MEDIADESCRIPTOR]
0000146C 72E1                     jb      short IOCtl_If7       ; Brif not

                                  ; 24/12/2023
                                  ;cmp    byte [disksector+26h], 29h ; [disksector+EXT_BOOT.SIG]
                                  ;                       ; EXT_BOOT_SIGNATURE
                                  ;jnz    short IOCtl_If7       ; not extended boot record

0000146E 06                       push    es             ; Save BDS pointer
0000146F 57                       push    di
00001470 1E                       push    ds             ; Point ES To boot record
00001471 07                       pop     es

                                  ; 24/12/2023
                                  ;;;
                                  ;mov    di, disksector+43h ; disksector+EXT_BOOT.SERIAL
00001472 BF[9401]                 mov     di, disksector+42h ; BS_FAT32_BootSig ; 24/12/2023
00001475 833E[6801]00             cmp     word [disksector+16h], 0 ; BPB.FATSz16
0000147A 7403                     jz      short IOCtl_If5       ; FAT32 fs
0000147C 83EF1C                   sub     di, 1Ch        ; 67-28 ; offset disksektor+27h
                                  ; di = disksector+26h = BS_BootSig ; 24/12/2023
                        IOCtl_If5:
                                  ;cmp    byte [di-1], 29h ; BS_BootSig or BS_FAT32_BootSig
0000147F 803D29                   cmp     byte [di], 29h
00001482 7404                     je      short IOCtl_If8
00001484 5F                       pop     di             ; not extended boot record
00001485 07                       pop     es
                                  ;jmp    short IOCtl_If7
                                  ; 24/12/2023
00001486 EBC6                     jmp     short IOCtl_If2
                        IOCtl_If8:
                                  ;;;
                                  ; 24/12/2023
                                  ;mov    di, disksector+27h ; disksector+EXT_BOOT.SERIAL
00001488 47                       inc     di
                                  ; di = disksector+27h (BS_VolID)
                                  ;      or disksector+43h (BS_FAT32_VolID)

00001489 C536[1200]               lds     si, [ptrsav]   ; ds:si points to request header.
0000148D C57413                   lds     si, [si+19]    ; [si+IOCTL_REQ.GENERICIOCTL_PACKET]
00001490 83C602                   add     si, 2          ; A_MEDIA_ID_INFO.MI_SERIAL

                                  ; 24/12/2023
                                  ;mov    cx, 23         ; size_of_EXT_BOOT_SERIAL
                                  ;                      ; +size_of_EXT_BOOT_VOL_LABEL
                                  ;                      ; +size_of_EXT_SYSTEM_ID
                                  ;rep movsb
00001493 E8B2FF                   call    IOCtl_If4       ; copy volume serial, label and system id

00001496 06                       push    es             ; point ds back to Bios_Data
00001497 1F                       pop     ds
00001498 5F                       pop     di             ; restore bds pointer
00001499 07                       pop     es
0000149A E8B2F3                   call    mov_media_ids  ; update the bds media id info.
0000149D 88D0                     mov     al, dl
0000149F C606[2001]03             mov     byte [rflag], 3       ; romwrite
000014A4 E80600                   call    BootIo         ; write it back.
000014A7 C606[1E01]FF             mov     byte [tim_drv], 0FFh
                                                         ; make sure chk_media check the driver
                                                         ; return with error code from BootIo
000014AC C3                       retn
                        ; ---------------------------------------------------------------------------

                                  ; 24/12/2023
                        ;IOCtl_If7:
                        ;         mov     al, 7          ; error_unknown_media
                        ;         stc
                        ;IOCtl_If6:
                        ;         retn

                        ; =============== S U B   R O U T I N E =======================================

                        ; 16/10/2022

                        ; ---------------------------------------------------------------------------
```

```
18102                                   ;   BootIo
18103                                   ; ---------------------------------------------------------------------------
18104                                   ;
18105                                   ; function: read/write the boot record into boot sector.
18106                                   ;
18107                                   ; input :
18108                                   ;       al=logical drive number
18109                                   ;       rFlag = operation (read/write)
18110                                   ;
18111                                   ; output:   for read operation,the boot record of the drive specified in bds
18112                                   ;       be read into the DiskSector buffer.
18113                                   ;       for write operation,the DiskSector buffer image will be written
18114                                   ;       to the drive specified in bds.
18115                                   ;       if carry set,then al contains the device driver error number
18116                                   ;       that will be returned to dos.
18117                                   ;       AX,CX,DX register destroyed.
18118                                   ;       if carry set,then al will contain the error code from DiskIO.
18119                                   ;
18120                                   ; subroutines to be called:
18121                                   ;   DiskIO:NEAR
18122                                   ;
18123                                   ; logic:
18124                                   ;
18125                                   ; {
18126                                   ;   first_sector = 0;       /*logical sector 0 is the boot sector */
18127                                   ;   sectorcount = 1;        /*read 1 sector only */
18128                                   ;   buffer = DiskSector;    /*read it into the DiskSector buffer */
18129                                   ;   call DiskIO (rFlag,drive_number,first_sector,sectorcount,buffer);
18130                                   ; }
18131                                   ; ===========================================================================
18132
18133                                             ; 19/10/2022
18134                                   BootIo:
18135 000014AD 06                                 push    es
18136 000014AE 57                                 push    di
18137 000014AF 53                                 push    bx
18138 000014B0 1E                                 push    ds
18139 000014B1 07                                 pop     es              ; Point ES: to Bios_Data
18140
18141                                             ; Call DiskIO to read/write the boot sec. The parameters which
18142                                             ; need to be initialized for this subroutine out here are
18143                                             ; - Transfer address to Bios_Data:DiskSector
18144                                             ; - Low sector needs to be initalized to 0. this is a reg. param
18145                                             ; - Hi sector in [Start_Sec_H] needs to be initialised to 0.
18146                                             ; - Number of sectors <-- 1
18147
18148 000014B2 BF[5201]                           mov     di, disksector ; es:di -> transfer address
18149 000014B5 31D2                               xor     dx, dx          ; First sector (h) -> 0
18150 000014B7 8916[9C04]                         mov     [start_sec_h], dx ; Start sector (h) -> 0
18151 000014BB B90100                             mov     cx, 1
18152 000014BE E850F5                             call    diskio
18153 000014C1 5B                                 pop     bx
18154 000014C2 5F                                 pop     di
18155 000014C3 07                                 pop     es
18156 000014C4 C3                                 retn
18157
18158                                   ; =============== S U B   R O U T I N E =======================================
18159
18160                                   ; 16/10/2022
18161
18162                                   ; ---------------------------------------------------------------------------
18163                                   ;   ChangeLineChk
18164                                   ; ---------------------------------------------------------------------------
18165                                   ;
18166                                   ; when the user calls get/set media id call before dos establishes the media
18167                                   ; by calling "media_chk",the change line activity of the drive is going to be
18168                                   ; lost.     this routine will check the change line activity and will save the
18169                                   ; history in the flags.
18170                                   ;
18171                                   ; FUNCTION: check the change line error activity
18172                                   ;
18173                                   ; INPUT :  ES:di -> bds table.
18174                                   ;
18175                                   ; OUTPUT:   flag in bds table will be updated if change line occurs.
18176                                   ;
18177                                   ; SUBROUTINES TO BE CALLED:
18178                                   ;   Set_Changed_DL
18179                                   ;
18180                                   ; ---------------------------------------------------------------------------
18181
18182                                             ; 24/12/2023 - Retro DOS 5.0
18183                                   ChangeLineChk:
18184 000014C5 268A5504                           mov     dl, [es:di+4]  ; [es:di+BDS.drivenum]
18185 000014C9 08D2                               or      dl, dl          ; Fixed disk?
18186 000014CB 7821                               js      short ChangeLnChkRet ; Yes, skip it.
18187                                             ; 24/12/2023
18188 000014CD 26F6453F04                         test    byte [es:di+3Fh], 4 ; [es:di+BDS.flags] ; PCDOS 7.1
18189                                             ; 12/12/2022
18190                                             ;test   byte [es:di+23h], 4
18191                                             ;;test  word [es:di+23h], 4 ; [es:di+BDS.flags]
18192                                                                     ; return_fake_bpb
18193 000014D2 751A                               jnz     short ChangeLnChkRet
18194 000014D4 803E[7700]01                       cmp     byte [fhave96], 1   ; This rom support change line?
18195 000014D9 7513                               jnz     short ChangeLnChkRet
18196 000014DB E8AD07                             call    haschange       ; This drive support change line?
18197 000014DE 740E                               jz      short ChangeLnChkRet ; Do nothing
18198
18199                                             ; Execute the rom disk interrupt to check changeline activity.
18200
18201 000014E0 B416                               mov     ah, 16h
18202 000014E2 CD13                               int     13h     ; DISK - FLOPPY       DISK - CHANGE OF DISK STATUS (AT,XT2,XT286,CONV,PS)
18203                                                             ; DL = drive to       check
18204                                                             ; Return: AH = disk change status
18205 000014E4 7308                               jnb     short ChangeLnChkRet
18206 000014E6 53                                 push    bx
18207 000014E7 BB4000                             mov     bx, 40h         ; fchanged
18208                                                                     ; Update flag in BDS for this
18209                                                                     ; physical drive
18210 000014EA E87707                             call    set_changed_dl
18211 000014ED 5B                                 pop     bx
18212                                   ChangeLnChkRet:
18213 000014EE C3                                 retn
18214
18215                                   ; ---------------------------------------------------------------------------
18216
18217                                   ; 16/10/2022
18218
18219                                   ; ===========================================================================
18220                                   ;   GetAccessFlag
18221                                   ; ===========================================================================
18222                                   ;
18223                                   ; FUNCTION: get the status of UNFORMATTED_MEDIA bit of flags in bds table
18224                                   ;
18225                                   ; INPUT :
```

```
18226                                   ;       ES:di -> bds table
18227                                   ;
18228                                   ; OUTPUT:   a_DiskAccess_Control.dac_access_flag = 0 if disk i/o not allowed.
18229                                   ;                                                  = 1 if disk i/o allowed.
18230                                   ; ============================================================================
18231
18232                                             ; 24/12/2023 - Retro DOS 5.0
18233
18234                                             ; 19/10/2022
18235                                   GetAccessFlag:
18236 000014EF C51E[1200]                         lds     bx, [ptrsav]   ; DS:BX points to request header
18237 000014F3 C55F13                             lds     bx, [bx+19]    ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET]
18238                                             ;mov    al, 0          ; Assume result       is unformatted
18239                                             ; 10/12/2022
18240 000014F6 28C0                               sub     al, al
18241                                             ; 24/12/2023
18242 000014F8 26F6454002                         test    byte [es:di+40h], 02h ; (PCDOS 7.1 IBMBIO.COM)
18243                                             ;test   word ptr es:[di+3Fh], 200h
18244                                             ; 10/12/2022
18245                                             ;test   byte [es:di+36], 02h
18246                                             ;;test  word [es:di+35], 200h ; [es:di+BDS.flags]
18247                                                                     ; unformatted_media
18248 000014FD 7501                               jnz     short GafDone  ; Done if unformatted
18249                                             ;inc    al             ; Return true for formatted
18250                                             ; 24/12/2023
18251 000014FF 40                                 inc     ax
18252                                   GafDone:
18253 00001500 884701                             mov     [bx+1], al     ; [bx+A_DISKACCESS_CONTROL.DAC_ACCESS_FLAG]
18254 00001503 C3                                 retn
18255                                   ; ---------------------------------------------------------------------------
18256
18257                                   ; 16/10/2022
18258
18259                                   ; ============================================================================
18260                                   ;   SetAccessFlag
18261                                   ; ============================================================================
18262                                   ;
18263                                   ; function: set/reset the UNFORMATTED_MEDIA bit of flags in bds table
18264                                   ;
18265                                   ; input :
18266                                   ;       ES:di -> bds table
18267                                   ;
18268                                   ; output:   unformtted_media bit modified according to the user request
18269                                   ; ============================================================================
18270
18271                                             ; 24/12/2023 - Retro DOS 5.0
18272
18273                                             ; 19/10/2022
18274                                   SetAccessFlag:
18275 00001504 C51E[1200]                         lds     bx, [ptrsav]   ; ES:BX points to request header
18276 00001508 C55F13                             lds     bx, [bx+19]    ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET]
18277                                             ; 24/12/2023
18278 0000150B 26806540FD                         and     byte [es:di+40h], 0FDh ; (PCDOS 7.1 IBMBIO.COM)
18279                                             ;and    word ptr es:[di+3Fh], 0FDFFh
18280                                             ; 10/12/2022
18281                                             ;and    byte [es:di+36], 0FDh
18282                                             ;;and   word [es:di+35], 0FDFFh ; [es:di+BDS.flags]
18283                                                                     ; ~unformatted_media
18284 00001510 807F0100                           cmp     byte [bx+1], 0 ; [bx+A_DISKACCESS_CONTROL.DAC_ACCESS_FLAG]
18285 00001514 7505                               jnz     short saf_Done
18286                                             ; 24/12/2023
18287 00001516 26804D4002                         or      byte [es:di+40h], 02h ; (PCDOS 7.1 IBMBIO.COM)
18288                                             ;or     word ptr es:[di+3Fh], 200h
18289                                             ; 15/04/2024
18290                                             ; 10/12/2022
18291                                             ;or     byte [es:di+36], 02h
18292                                             ;;or    word [es:di+35], 200h ; [es:di+BDS.flags]
18293                                                                     ; unformatted_media
18294                                   saf_Done:
18295 0000151B C3                                 retn
18296                                   ; ---------------------------------------------------------------------------
18297
18298                                   ; 16/10/2022
18299
18300                                   ; ============================================================================
18301                                   ; Ioctl_Support_Query
18302                                   ; ============================================================================
18303                                   ;
18304                                   ; New device command which was added in DOS 5.00 to allow a query of a
18305                                   ; specific GENERIC IOCtl to see if it is supported. Bit 7 in the
18306                                   ; device attributes specifies if this function is supported.
18307                                   ;
18308                                   ; ============================================================================
18309
18310                                             ; 24/12/2023 - Retro DOS 5.0
18311
18312                                             ; 19/10/2022
18313                                   ioctl_support_query:
18314 0000151C 06                                 push    es
18315 0000151D C41E[1200]                         les     bx, [ptrsav]   ; ES:BX Points to request header.
18316 00001521 268B470D                           mov     ax, [es:bx+13] ; [es:bx+IOCTL_REQ.MAJORFUNCTION]
18317                                                                     ; AL == Major, AH == Minor
18318                                             ; 24/12/2023
18319                                             ; 02/09/2023 (PCDOS 7.1)
18320 00001525 3C48                               cmp     al, 48h        ; IOC_NEW_DC (PCDOS 7.1)
18321                                                                     ; new generic ioctl function (FAT32)
18322 00001527 7404                               je      short ioctl_support
18323
18324 00001529 3C08                               cmp     al, 8          ; IOC_DC
18325                                                                     ; See if major code is 8
18326 0000152B 7513                               jne     short nosupport
18327                                   ioctl_support:
18328 0000152D 0E                                 push    cs
18329 0000152E 07                                 pop     es
18330                                             ; 24/12/2023
18331                                             ; 02/09/2023
18332 0000152F B90E00                             mov     cx, 14          ; (PCDOS 7.1) IOC_DC_TABLE_LEN
18333                                             ;mov    cx, 11         ; IOC_DC_TABLE_LEN
18334                                             ; 10/12/2022
18335 00001532 BF[BD0E]                           mov     di, IOC_DC_Table
18336                                             ;mov    di, 0C60h      ; IOC_DC_Table
18337                                                                     ; at 2C7h:0C60h       = 70h:31D0h
18338 00001535 86E0                               xchg    al, ah         ; Put minor code in AL
18339 00001537 F2AE                               repne scasb            ; Scan for minor code in AL
18340 00001539 7505                               jnz     short nosupport       ; it was not found
18341 0000153B B80001                             mov     ax, 100h
18342                                             ; 10/12/2022
18343                                             ; (jump to ioctlsupexit is not required)
18344                                             ;jmp    short $+2      ; ioctlsupexit
18345                                                                     ; Signal ioctl is supported
18346                                             ;;jmp   short ioctlsupexit
18347                                   ; ---------------------------------------------------------------------------
18348                                   ioctlsupexit:
18349 0000153E 07                                 pop     es
```

```
18350                                           ; 10/12/2022
18351                                           ; cf = 0
18352                                           ;clc
18353 0000153F C3                               retn
18354                                   ; ---------------------------------------------------------------------------
18355                                   nosupport:
18356 00001540 07                               pop     es
18357 00001541 E991EB                           jmp     bc_cmderr
18358                                   ; ---------------------------------------------------------------------------
18359
18360                                   ; 16/10/2022
18361
18362                                   ; ===========================================================================
18363                                   ;   GetMediaSenseStatus
18364                                   ; ===========================================================================
18365                                   ;
18366                                   ; FUNCTION: Will return the type of diskette media in the specified DOS
18367                                   ;       diskette drive and whether the media is the default type
18368                                   ;       for that drive. (default type means the max size for that
18369                                   ;       drive)
18370                                   ;
18371                                   ; INPUT :   ES:DI -> BDS table
18372                                   ; OUTPUT:   If carry clear
18373                                   ;       DS:BX -> Updated IOCtlPacket
18374                                   ;
18375                                   ;                   Special Function at offset 0:
18376                                   ;                         0       - Media detected is not default type
18377                                   ;                         1       - Media detected is default type
18378                                   ;
18379                                   ;                   Device Type at offset 1:
18380                                   ;                         2       - 720K 3.5" 80 tracks
18381                                   ;                         7       - 1.44M 3.5" 80 tracks
18382                                   ;                         9       - 2.88M 3.5" 80 tracks
18383                                   ;
18384                                   ; Error Codes returned in AX if carry set:
18385                                   ;
18386                                   ; 8102 - Drive not ready  - No disk is in the drive.
18387                                   ; 8107 - Unknown media type       - Drive doesn't support this function or
18388                                   ;                           the media is really unkown, any error
18389                                   ;                           other than "media not present"
18390                                   ;
18391                                   ; ===========================================================================
18392
18393                                           ; 19/10/2022
18394                                   SenseMediaType:
18395 00001544 C51E[1200]                       lds     bx, [ptrsav]   ; DS:BX points to request header.
18396 00001548 C55F13                           lds     bx, [bx+19]    ; bx+IOCTL_REQ.GENERICIOCTL_PACKET]
18397                                           ; 10/10/2022
18398                                           ;mov    word [bx], 0   ; Initialize the 2 packet bytes
18399 0000154B 31D2                             xor     dx, dx
18400 0000154D 8917                             mov     [bx], dx ; 0
18401                                           ;
18402 0000154F 268A5504                         mov     dl, [es:di+4]  ; [es:di+BDS.drivenum]
18403                                                                  ; Get int 13h drive number from      BDS
18404                                           ; 10/12/2022
18405                                           ;xor    dh, dh         ; DX = physical       drive number
18406 00001553 B420                             mov     ah, 20h        ; Get Media Type function
18407                                                                  ; If no carry media type in AL
18408 00001555 CD13                             int     13h            ; DISK - QCACHE       - DISMOUNT
18409 00001557 7216                             jc      short MediaSenseEr ; error code      in AH
18410 00001559 FE07                             inc     byte [bx]      ; Signal media type is default (bit 1)
18411                                   DetermineMediaType:
18412 0000155B FEC8                             dec     al
18413 0000155D 3C02                             cmp     al, 2          ; Chk for 720K ie: (3-1) = 2
18414 0000155F 740A                             jz      short GotMediaType
18415 00001561 0404                             add     al, 4
18416 00001563 3C07                             cmp     al, 7          ; Chk for 1.44M ie: (4-1+4) = 7
18417 00001565 7404                             jz      short GotMediaType
18418 00001567 3C09                             cmp     al, 9          ; Chk for 2.88M       ie: (6-1+4) = 9
18419 00001569 7510                             jnz     short UnknownMediaType ; Just didn't recognize media type
18420                                   GotMediaType:
18421 0000156B 884701                           mov     [bx+1],al      ; Save the return value
18422                                           ; 10/12/2022
18423                                           ; cf = 0
18424                                           ;clc                   ; Signal success
18425 0000156E C3                               retn
18426                                   ; ---------------------------------------------------------------------------
18427
18428                                   MediaSenseEr:
18429 0000156F 80FC32                           cmp     ah, 32h        ; See if not default media error
18430 00001572 74E7                             jz      short DetermineMediaType ; Not really an error
18431 00001574 B002                             mov     al, 2          ; Now assume drive not ready
18432 00001576 80FC31                           cmp     ah, 31h        ; See if media was present
18433 00001579 7402                             jz      short SenseErrExit ; Return drive not ready
18434                                   UnknownMediaType:
18435 0000157B B007                             mov     al, 7          ; Just don't know the media type
18436                                   SenseErrExit:
18437 0000157D B481                             mov     ah, 81h        ; Signal error return
18438 0000157F F9                               stc
18439 00001580 C3                               retn
18440
18441                                   ; ---------------------------------------------------------------------------
18442                                           ; 10/12/2022
18443                                           ;db    0
18444                                   ; ---------------------------------------------------------------------------
18445
18446                                   ;---------------------------------------------------------------------------
18447                                   ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:15F2h
18448                                   ;---------------------------------------------------------------------------
18449                                   ; 26/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
18450
18451                                   ; =============== S U B R O U T I N E =======================================
18452
18453                                   SetLockState:
18454 00001581 C51E[1200]                       lds     bx, [ptrsav]   ; set media lock state
18455 00001585 C55F13                           lds     bx, [bx+13h]   ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET]
18456                                           ;mov    dl, [es:di+4]  ; [es:di+BDS.drivenum]
18457                                           ;call   check_int13h_exts_present
18458                                           ; 26/12/2023
18459 00001588 E82100                           call    check_int13h_exts_p
18460                                           ;mov    al, 3          ; unknown command error
18461 0000158B 721C                             jc      short setlockst_ret
18462 0000158D 8A07                             mov     al, [bx]       ; [bx+A_LOCKSTATE_CONTROL.LOCKSTATE_FUNCTIONS]
18463 0000158F B445                             mov     ah, 45h
18464 00001591 CD13                             int     13h            ; DISK - IBM/MS Extension - LOCK/UNLOCK DRIVE
18465                                           ; (DL - drive, [SI - disk address packet)
18466 00001593 884701                           mov     [bx+1], al     ; 1 = locked, 0 = not locked
18467                                           ; [bx+A_LOCKSTATE_CONTROL.LOCKSTATE_FLAG]
18468
18469                                           ; 26/12/2023
18470 00001596 EB0A                             jmp     short sls_em
18471
18472                                   ;       jnc     short setlockst_ret
18473                                   ;       mov     al, ah
```

```
18474                                   ;           call    maperror
18475                                   ;setlockst_ret:
18476                                   ;           mov     ah, 81h         ; Return this status in case of carry
18477                                   ;           retn
18478
18479                                   ; =============== S U B R O U T I N E =======================================
18480
18481                                   EjectMedia:
18482                                               ;mov    dl, [es:di+4]  ; eject media in drive
18483                                               ;                      ; [es:di+BDS.drivenum]
18484                                               ;call   check_int13h_exts_present
18485                                               ; 26/12/2023
18486 00001598 E81100                             call    check_int13h_exts_p
18487                                               ;mov    al, 3          ; unknown command error
18488 0000159B 720C                               jc      short ejectm_ret
18489 0000159D B80046                             mov     ax, 4600h
18490 000015A0 CD13                               int     13h             ; DISK - IBM/MS Extension - EJECT MEDIA
18491                                               ; (DL - drive)
18492                                   sls_em:                     ; 26/12/2023
18493 000015A2 7305                               jnc     short ejectm_ret
18494 000015A4 88E0                               mov     al, ah
18495 000015A6 E800F8                             call    maperror
18496                                   setlockst_ret:              ; 26/12/2023
18497                                   ejectm_ret:
18498 000015A9 B481                               mov     ah, 81h         ; Return this status in case of carry
18499 000015AB C3                                 retn
18500
18501                                   ; =============== S U B R O U T I N E =======================================
18502
18503                                               ; 26/12/2023
18504                                   check_int13h_exts_p:
18505 000015AC 268A5504                           mov     dl, [es:di+4]
18506
18507                                   check_int13h_exts_present:
18508 000015B0 B441                               mov     ah, 41h
18509 000015B2 53                                 push    bx
18510 000015B3 BBAA55                             mov     bx, 55AAh
18511 000015B6 CD13                               int     13h             ; DISK - Check for INT 13h Extensions
18512                                                                     ; BX = 55AAh, DL = drive number
18513                                                                     ; Return: CF set if not supported
18514                                                                     ; AH = extensions version
18515                                                                     ; BX = AA55h
18516                                                                     ; CX = Interface support bit map
18517 000015B8 81FB55AA                                   cmp     bx, 0AA55h
18518 000015BC 5B                                         pop     bx
18519 000015BD 7505                                       jnz     short exts_notsupported
18520 000015BF F6C102                                     test    cl, 2           ; bit 1 - drive locking and ejecting subset
18521 000015C2 7503                                       jnz     short exts_supported
18522                                   exts_notsupported:
18523                                               ; 26/12/2023
18524 000015C4 B003                               mov     al, 3
18525                                               ;
18526 000015C6 F9                                 stc
18527                                   exts_supported:
18528 000015C7 C3                                 retn
18529
18530                                   ; =============== S U B R O U T I N E =======================================
18531
18532                                   GetDrvMapInfo:
18533 000015C8 8CD9                               mov     cx, ds          ; get drive map information
18534                                                                     ;
18535                                                                     ; es:di points to BDS which belongs to
18536                                                                     ;       the requested logical/dos drive number
18537                                                                     ;
18538                                                                     ; Format of parameter block:
18539                                                                     ; Offset  Description (Table 01570)
18540                                                                     ;  00h    (call) length of this buffer (in bytes)
18541                                                                     ;  01h    (ret) number of bytes in parameter block
18542                                                                     ;           actually used
18543                                                                     ;  02h    (ret) drive flags
18544                                                                     ;  03h    (ret) physical drive number
18545                                                                     ;           00h-7Fh floppy
18546                                                                     ;           80h-FEh hard
18547                                                                     ;           FFh no physical drive
18548                                                                     ;  04h    (ret) bitmap of logical drives associated with
18549                                                                     ;           physical drive
18550                                                                     ;           bit 0 = drive A:, etc.
18551                                                                     ;  08h    (ret) relative block address of partition start
18552                                                                     ;           qword
18553                                                                     ;
18554                                                                     ; Ref: Ralf Brown's Interrupt List, INTERRUP.G
18555 000015CA C51E[1200]                                 lds     bx, [ptrsav]
18556 000015CE C55F13                                     lds     bx, [bx+13h]    ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET]
18557 000015D1 B80381                                     mov     ax, 8103h       ; ah = generic ioctl error code (81h)
18558                                                                             ; al = unknown command error (03h)
18559 000015D4 803F10                                     cmp     byte [bx], 10h ; parameter buffer length = 16 bytes
18560 000015D7 7251                                       jb      short gdmi_4
18561 000015D9 268A5504                                   mov     dl, [es:di+4]  ; [es:di+BDS.drivenum]
18562 000015DD 885703                                     mov     [bx+3], dl     ; parameter block - offset 3 - physical drive number
18563 000015E0 C6470110                                   mov     byte [bx+1], 10h ; parameter block - actually used length
18564 000015E4 268B4517                                   mov     ax, [es:di+17h]       ; [es:di+BDS.hiddensectors]
18565 000015E8 894708                                     mov     [bx+8], ax     ; parameter block - offset 8 - partition start LBA
18566 000015EB 268B4519                                   mov     ax, [es:di+19h]       ; [es:di+BDS.hiddensectors+2]
18567 000015EF 89470A                                     mov     [bx+0Ah], ax   ; parameter block - offset 10
18568 000015F2 31C0                                       xor     ax, ax ; 0
18569 000015F4 884702                                     mov     [bx+2], al     ; drive flags = 0 (protected mode flags etc.)
18570 000015F7 89470C                                     mov     [bx+0Ch], ax   ; high dword of partition start address (LBA) is 0
18571 000015FA 89470E                                     mov     [bx+0Eh], ax
18572 000015FD 894704                                     mov     [bx+4], ax     ; logical drive bitmap of same physical drive
18573                                                                             ; initialized as 0
18574 00001600 894706                                     mov     [bx+6], ax ; 0
18575 00001603 8EC1                                       mov     es, cx
18576                                                     ;les    di, dword ptr es:start_bds ; 1st BDS
18577 00001605 26C43E[1901]                               les     di, [es:start_bds]
18578 0000160A B90100                                     mov     cx, 1          ; bit 0 (drive A:)
18579                                   gdmi_1:
18580 0000160D 83FFFF                                     cmp     di, 0FFFFh     ; last BDS ?
18581 00001610 7415                                       jz      short gdmi_3   ; yes
18582 00001612 26385504                                   cmp     [es:di+4], dl  ; [es:di+BDS.drivenum], dl
18583                                                                             ; is it same physical drive ?
18584 00001616 7506                                       jnz     short gdmi_2   ; no
18585 00001618 094F04                                     or      [bx+4], cx     ; set bit for logical drive index of this BDS
18586                                                                             ; (previously) shifted bit (which is 1/ON) is in ax:cx
18587 0000161B 094706                                     or      [bx+6], ax
18588                                   gdmi_2:
18589 0000161E D1E1                                       shl     cx, 1          ; shift one left for setting the next drive's bit
18590 00001620 D1D0                                       rcl     ax, 1          ; set high word of the bit select (set) value
18591 00001622 26C43D                                     les     di, [es:di]    ; next BDS
18592 00001625 EBE6                                       jmp     short gdmi_1   ; loop until di = -1 (last BDS sign)
18593                                   gdmi_3:
18594 00001627 B80001                                     mov     ax, 100h       ; success
18595                                   gdmi_4:
18596 0000162A C3                                 retn
18597
```

```
                        ;--------------------------------------------------------------------------

                        ; 16/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                        ; 26/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)

                        ;--------------------------------------------------------------------------
                        ; MSINT13.ASM - MSDOS 6.0 - 1991
                        ;--------------------------------------------------------------------------
                        ; 16/03/2019 - Retro DOS v4.0

                        ;   int 2f function 13h allows the user to change the orig13 int_13 vector
                        ;   after booting. this allows testing and implementation of custom int_13
                        ;   handlers, without giving up ms-dos error recovery
                        ;   entry:  ds:dx   == addr. of new int_13 handler
                        ;           es:bx   == addr. of new int_13 vector used by warm boot (int19)
                        ;   exit:   orig13  == address of new int_13 handler
                        ;           ds:dx   == old orig13 value
                        ;           es:bx   == old old13  value
                        ;
                        ; int 2f handler for external block drivers to communicate with the internal
                        ; block driver in msdisk. the multiplex number chosen is 8. the handler
                        ; sets up the pointer to the request packet in [ptrsav] and then jumps to
                        ; dsk_entry, the entry point for all disk requests.
                        ;
                        ; on exit from this driver, we will return to the external driver
                        ; that issued this int 2f, and can then remove the flags from the stack.
                        ; this scheme allows us to have a small external device driver, and makes
                        ; the maintainance of the various drivers (driver and msbio) much easier,
                        ; since we only need to make changes in one place (most of the time).
                        ;
                        ;   ax=800h - check for installed handler - reserved
                        ;   ax=801h - install the bds into the linked list
                        ;   ax=802h - dos request
                        ;   ax=803h - return bds table starting pointer in ds:di
                        ;      (ems device driver hooks int 13h to handle 16kb dma overrun
                        ;       problem. bds table is going to be used to get head/sector
                        ;       informations without calling generic ioctl get device parm call.)

                        ;BIOSSEGMENT equ 70h
                        DOSBIOSSEG equ 0070h ; 17/10/2022

                        ;;BIOSCODE:1302h (MSDOS 6.21, IO.SYS)
                        ;BIOSCODE:16AAh (PCDOS 7.1, IBMBIO.COM) ; 26/12/2023

                        i2f_handler:                            ; here is 02C7h:1302h =      0070h:3872h
0000162B 80FC13                 cmp     ah, 13h
0000162E 7413                   jz      short int2f_replace_int13
00001630 80FC08                 cmp     ah, 8
00001633 7432                   jz      short mine

                        ; Check for WIN386 startup and return the BIOS instance data

00001635 80FC16                 cmp     ah, 16h         ; MultWin386
00001638 746D                   jz      short win386call
0000163A 80FC4A                 cmp     ah, 4Ah         ; multMULT
0000163D 7503                   jnz     short i2f_handler_iret
0000163F E99800                 jmp     handle_multmult
                        ; --------------------------------------------------------------------------

                        i2f_handler_iret:
00001642 CF                     iret
                        ; --------------------------------------------------------------------------

                        int2f_replace_int13:
00001643 FA                     cli     ; 26/12/2023
00001644 50                     push    ax      ; free up a register for caller's ds
00001645 8CD8                   mov     ax, ds  ; then we can use ds: -> Bios_Data
                                ;;mov   ds, word [cs:0030h] ; 15/10/2022
                                ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:0030h]
                                                ; = [02C7h:0030h] = [0070h:25A0h]
00001647 2E8E1E[3000]           mov     ds, [cs:BIOSDATAWORD] ; 17/10/2022
                                ; 19/10/2022
                                ;push   word ptr ds:Orig13    ; save old value of old13 and
                                ;push   word ptr ds:Orig13+2  ; orig13 so that we can
                                ;push   word ptr ds:Old13     ; return them to caller
                                ;push   word ptr ds:Old13+2

                                ; 02/09/2023 (PCDOS 7.1)
                                ;push   word [Orig13]
0000164C FF36[B600]             push    word [Orig13+2]
                                ;push   word [Old13]
00001650 FF36[0801]             push    word [Old13+2]

                                ;mov    word ptr ds:Orig13, dx; orig13 := addr. of new int_13
                                ;mov    word ptr ds:Orig13+2, ax
                                ;mov    word ptr ds:Old13, bx ; old13 := addr. of new boot_13
                                ;mov    word ptr ds:Old13+2, es

                                ;mov    [Orig13], dx
                                ; 02/09/2023
00001654 8716[B400]             xchg    dx, [Orig13]
00001658 A3[B600]               mov     [Orig13+2], ax
                                ;mov    [Old13], bx
                                ; 02/09/2023
0000165B 871E[0601]             xchg    bx, [Old13]
0000165F 8C06[0801]             mov     [Old13+2], es

00001663 07                     pop     es                      ; es:bx := old old13 vector
                                ; 02/09/2023
                                ;pop    bx
00001664 1F                     pop     ds                      ; ds:dx := old orig13 vector
                                ;pop    dx ; 02/09/2023
00001665 58                     pop     ax
                        i2f_iret:
00001666 CF                     iret
                        ; --------------------------------------------------------------------------

                        mine:
00001667 3CF8                   cmp     al, 0F8h                ; iret on reserved functions
00001669 73FB                   jnb     short i2f_iret
0000166B 08C0                   or      al, al                  ; a get installed state request?
0000166D 7503                   jnz     short disp_func
0000166F B0FF                   mov     al, 0FFh
                                ;jmp    short i2f_iret
                                ; 02/09/2023
00001671 CF                     iret
                        ; --------------------------------------------------------------------------

                        disp_func:
00001672 3C01                   cmp     al, 1                   ; request for installing bds?
00001674 7418                   jz      short do_subfun_01
00001676 3C03                   cmp     al, 3                   ; get bds vector?
00001678 7423                   jz      short do_get_bds_vector
```

```
                                ; set up pointer to request packet
                                
0000167A 1E                             push    ds
0000167B 2E8E1E[3000]                   mov     ds, [cs:BIOSDATAWORD] ; 17/10/2022
                                        ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
                                                                ; = [0070h:25A0h] = [02C7h:0030h]
                                        ; 19/10/2022
                                        ;mov    word ptr ds:ptrsav, bx
                                        ;mov    word ptr ds:ptrsav+2, es
00001680 891E[1200]                     mov     [ptrsav], bx
00001684 8C06[1400]                     mov     [ptrsav+2], es
00001688 1F                             pop     ds
                                        ;jmp    far ptr i2f_dskentry
                                        ; 07/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                        ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:1708h
00001689 EA[5E06]7000                   jmp     DOSBIOSSEG:dsk_entry ; BIOSDATA:dsk_entry
                                        ;; 17/10/2022
                                        ;;jmp   far DOSBIOSSEG:dsk_entry
                                        ;jmp    DOSBIOSSEG:i2f_dskentry ; 70h:i2f_dskentry
                                                                ; NOTE: jump to a FAR function, not an
                                                                ;  IRET type function. Callers of
                                                                ;  this int2f subfunction will have
                                                                ;  to be careful to do a popf
                                
                                ; ---------------------------------------------------------------------------
                                
                                do_subfun_01:
0000168E 06                             push    es
0000168F 1E                             push    ds
00001690 1E                             push    ds
00001691 07                             pop     es
                                        ; 17/10/2022
00001692 2E8E1E[3000]                   mov     ds, [cs:BIOSDATAWORD]
                                        ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
                                                                ; point ds: -> Bios_Data
00001697 E8BC03                         call    install_bds
0000169A 1F                             pop     ds
0000169B 07                             pop     es
                                        ;jmp    short i2f_iret
                                        ; 02/09/2023
0000169C CF                             iret
                                ; ---------------------------------------------------------------------------
                                
                                do_get_bds_vector:
                                        ; 17/10/2022
0000169D 2E8E1E[3000]                   mov     ds, [cs:BIOSDATAWORD]
                                        ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
000016A2 C53E[1901]                     lds     di, [start_bds]
                                        ;lds    di, ds:start_bds
                                ;ii2f_iret: ; 10/12/2022
                                        ;jmp    short i2f_iret
                                        ; 02/09/2023
000016A6 CF                             iret
                                ; ---------------------------------------------------------------------------
                                
                                ; 17/10/2022
                                ; 16/10/2022
                                
                                ; WIN386 startup stuff is done here. If starting up we set our WIN386 present
                                ; flag and return instance data. If exiting, we reset the WIN386 present flag
                                ; NOTE: We assume that the BIOS int 2fh is at the bottom of the chain.
                                
                                win386call:
000016A7 1E                             push    ds
000016A8 2E8E1E[3000]                   mov     ds, [cs:BIOSDATAWORD]
                                        ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
                                                                ; at 2C7h:30h =       70h:25A0h
000016AD 3C05                           cmp     al, 5           ; Win386_Init
                                                                ; is itwin386 initializing?
000016AF 7410                           jz      short Win386Init
000016B1 3C06                           cmp     al, 6           ; Win386_Exit
                                                                ; is itwin386 exiting?
000016B3 7523                           jnz     short win_iret ; if not, continue int2f chain
                                        ; 12/12/2022
000016B5 F6C201                         test    dl, 1
                                        ;test   dx, 1           ; is itwin386 or win286 dos extender?
000016B8 751E                           jnz     short win_iret ; if not win386, then continue
                                        ;and    ds:IsWin386, 0 ; indicate that       win386 is not present
000016BA 8026[1208]00                   and     byte [IsWin386], 0
000016BF EB17                           jmp     short win_iret
                                ; ---------------------------------------------------------------------------
                                
                                Win386Init:
                                        ; 12/12/2022
000016C1 F6C201                         test    dl, 1
                                        ;test   dx, 1           ; is it win386 or win286 dos extender?
000016C4 7512                           jnz     short win_iret ; if not win386, then continue
                                        ;or     ds:IsWin386, 1 ; Indicate WIN386 present
000016C6 800E[1208]01                   or      byte [IsWin386], 1
                                        ;mov    word ptr ds:SI_Next, bx       ; Hook our structure into chain
                                        ;mov    word ptr ds:SI_Next+2, es
000016CB 891E[E007]                     mov     [SI_Next], bx
000016CF 8C06[E207]                     mov     [SI_Next+2], es
                                        ;mov    bx, offset Win386_SI ; point ES:BX to Win386_SI
000016D3 BB[DE07]                       mov     bx, Win386_SI  ; 19/10/2022
000016D6 1E                             push    ds
000016D7 07                             pop     es
                                win_iret:
000016D8 1F                             pop     ds
                                ii2f_iret:  ; 10/12/2022
                                        ;jmp    short i2f_iret ; return back up the chain
                                        ; 02/09/2023
000016D9 CF                             iret
                                ; ---------------------------------------------------------------------------
                                
                                handle_multmult:
000016DA 3C01                           cmp     al, 1
000016DC 7514                           jnz     short try_2
000016DE 1E                             push    ds
000016DF E84500                         call    HMAPtr          ; get offset of free HMA
                                        ; 10/12/2022
                                        ;xor    bx, bx
                                        ;dec    bx
000016E2 BBFFFF                         mov     bx, 0FFFFh
000016E5 8EC3                           mov     es, bx          ; seg of HMA
000016E7 89FB                           mov     bx, di
000016E9 F7D3                           not     bx
000016EB 09DB                           or      bx, bx
000016ED 7401                           jz      short try_1
000016EF 43                             inc     bx
                                try_1:
000016F0 1F                             pop     ds
                                        ;jmp    short ii2f_iret
                                        ; 02/09/2023
```

```
18846 000016F1 CF                       iret
18847                           ; ---------------------------------------------------------------------------
18848
18849                           try_2:
18850 000016F2 3C02                     cmp     al, 2           ; multMULTALLOCHMA
18851 000016F4 7530                     jnz     short try_3
18852 000016F6 1E                       push    ds
18853                                   ; 10/12/2022
18854                                   ;xor    di, di
18855                                   ;dec    di
18856 000016F7 BFFFFF                   mov     di, 0FFFFh      ; assume not enough space
18857 000016FA 8EC7                     mov     es, di
18858 000016FC E82800                   call    HMAPtr          ; get offset of free HMA
18859 000016FF 83FFFF                   cmp     di, 0FFFFh
18860 00001702 7421                     jz      short InsuffHMA
18861 00001704 F7DF                     neg     di              ; free space in HMA
18862 00001706 39FB                     cmp     bx, di
18863 00001708 7605                     jbe     short try_4
18864                                   ; 10/12/2022
18865                                   ;sub    di, di
18866                                   ;dec    di
18867 0000170A BFFFFF                   mov     di, 0FFFFh
18868                                   ;jmp    short InsuffHMA
18869                                   ; 02/09/2023
18870 0000170D 1F                       pop     ds
18871 0000170E CF                       iret
18872                           ; ---------------------------------------------------------------------------
18873
18874                           try_4:
18875                                   ;mov    di, ds:FreeHMAPtr
18876 0000170F 8B3E[D707]               mov     di, [FreeHMAPtr]
18877 00001713 83C30F                   add     bx, 15
18878                                   ;and    bx, 0FFF0h
18879                                   ; 10/12/2022
18880 00001716 80E3F0                   and     bl, 0F0h
18881                                   ;add    ds:FreeHMAPtr, bx ; update the free pointer
18882 00001719 011E[D707]               add     [FreeHMAPtr], bx
18883 0000171D 7506                     jnz     short InsuffHMA
18884 0000171F C706[D707]FFFF           mov     word [FreeHMAPtr], 0FFFFh ; -1
18885                                   ;mov    ds:FreeHMAPtr, 0FFFFh
18886                                                   ; no more HMA if we have wrapped
18887                           InsuffHMA:
18888 00001725 1F                       pop     ds
18889                                   ; 10/12/2022
18890                           try_3:
18891                                   ;jmp    short ii2f_iret
18892                                   ; 02/09/2023
18893 00001726 CF                       iret
18894                           ; ---------------------------------------------------------------------------
18895
18896                                   ; 10/12/2022
18897                           ;try_3:
18898                                   ;jmp    ii2f_iret
18899
18900                           ; =============== S U B   R O U T I N E =======================================
18901
18902                           ; 16/10/2022
18903
18904                           ;---------------------------------------------------------------------------
18905                           ;
18906                           ; procedure : HMAPtr
18907                           ;
18908                           ;           Gets the offset of the free HMA area ( with respect to
18909                           ;                                                  seg ffff )
18910                           ;           If DOS has not moved high, tries to move DOS high.
18911                           ;           In the course of doing this, it will allocate all the HMA
18912                           ;           and set the FreeHMAPtr to past the end of the BIOS and
18913                           ;           DOS code. The call to MoveDOSIntoHMA (which is a pointer)
18914                           ;           enters the routine in sysinit1 called FTryToMovDOSHi.
18915                           ;
18916                           ;   RETURNS : offset of free HMA in DI
18917                           ;             BIOS_DATA, seg in DS
18918                           ;
18919                           ;---------------------------------------------------------------------------
18920
18921                                   ; 17/10/2022
18922                           HMAPtr:
18923 00001727 2E8E1E[3000]             mov     ds, [cs:BIOSDATAWORD]
18924                                   ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
18925 0000172C 8B3E[D707]               mov     di, [FreeHMAPtr]
18926                                   ;mov    di, ds:FreeHMAPtr
18927 00001730 83FFFF                   cmp     di, 0FFFFh
18928 00001733 750F                     jnz     short HMAPtr_retn
18929 00001735 803E[DD07]00             cmp     byte [SysinitPresent], 0
18930                                   ;cmp    ds:SysinitPresent, 0
18931 0000173A 7408                     jz      short HMAPtr_retn
18932 0000173C FF1E[D907]               call    far [MoveDOSIntoHMA]
18933                                   ;call   ds:MoveDOSIntoHMA ; call far [MoveDOSIntoHMA]
18934 00001740 8B3E[D707]               mov     di, [FreeHMAPtr]
18935                                   ;mov    di, ds:FreeHMAPtr
18936                           HMAPtr_retn:
18937 00001744 C3                       retn
18938
18939                           ; =============== S U B   R O U T I N E =======================================
18940
18941                           ; 16/10/2022
18942
18943                           ; move a 512 byte sector from ds:si to es:di, do not trash cx
18944                           ; but go ahead and update direction flag, si, & di
18945
18946                           move_sector:
18947
18948                           ; The 80386 microprocessor considers an access to WORD 0FFFFh in
18949                           ; any segment to be a fault. Theoretically, this could be handled
18950                           ; by the fault handler and the behavior of an 8086 could be emulated
18951                           ; by wrapping the high byte to offset 0000h. This would be a lot
18952                           ; of work and was, indeed, blown off by the Win386 guys. COMPAQ
18953                           ; also handles the fault incorrectly in their ROM BIOS for real
18954                           ; mode. Their fault handler was only designed to deal with one
18955                           ; special case which occurred in a magazine benchmark, but didn't
18956                           ; handle the general case worth beans.
18957                           ;
18958                           ; Simply changing this code to do a byte loop would work okay but
18959                           ; would involve a general case performance hit. Therefore, we'll
18960                           ; check for either source or destination offsets being within one
18961                           ; sector of the end of their segments and only in that case fall
18962                           ; back to a byte move.
18963
18964 00001745 FC                       cld
18965 00001746 51                       push    cx
18966 00001747 B90001                   mov     cx, 256
18967 0000174A 81FE00FE                 cmp     si, 0FE00h
18968 0000174E 770A                     ja      short movsec_bytes
18969 00001750 81FF00FE                 cmp     di, 0FE00h
```

```
18970 00001754 7704                     ja      short movsec_bytes
18971 00001756 F3A5                     rep movsw
18972 00001758 59                       pop     cx
18973 00001759 C3                       retn
18974                           ; ---------------------------------------------------------------------------
18975
18976                           movsec_bytes:
18977 0000175A D1E1                     shl     cx, 1
18978 0000175C F3A4                     rep movsb
18979 0000175E 59                       pop     cx
18980 0000175F C3                       retn
18981
18982                           ; =============== S U B   R O U T I N E =======================================
18983
18984                           ; 16/10/2022
18985
18986                           ; check_wrap is a routine that adjusts the starting sector, starting head
18987                           ; and starting cylinder for an int 13 request that requests i/o of a lot
18988                           ; of sectors. it only does this for fixed disks. it is used in the sections
18989                           ; of code that handle ecc errors and dma errors. it is necessary, because
18990                           ; ordinarily the rom would take care of wraps around heads and cylinders,
18991                           ; but we break down a request when we get an ecc or dma error into several
18992                           ; i/o of one or more sectors. in this case, we may already be beyond the
18993                           ; number of sectors on a track on the medium, and the request would fail.
18994                           ;
18995                           ; input conditions:
18996                           ;   all registers set up for an int 13 request.
18997                           ;
18998                           ; output:
18999                           ;   dh - contains starting head number for request
19000                           ;   cx - contains starting sector and cylinder numbers
19001                           ;   (the above may or may not have been changed, and are 0-based)
19002                           ;   all other registers preserved.
19003
19004                                   ; 26/12/2023 - Retro DOS 5.0
19005                           check_wrap:
19006 00001760 50                       push    ax
19007 00001761 53                       push    bx
19008 00001762 06                       push    es
19009 00001763 57                       push    di
19010 00001764 E86C00                   call    find_bds        ; get pointer to bds for drive in dl
19011 00001767 725E                     jb      short no_wrap   ; finished if DOS doesn't use it
19012                                   ; 26/12/2023
19013 00001769 26F6453F01               test    byte [es:di+3Fh], 1
19014                                   ; 12/12/2022
19015                                   ;test   byte [es:di+23h], 1
19016                                   ;;test  word [es:di+23h], 1 ; [es:di+BDS.flags],fnon_removable
19017 0000176E 7457                     jz      short no_wrap   ; no wrapping for removable media
19018 00001770 268B5D13                 mov     bx, [es:di+13h]         ; [es:di+BDS.secpertrack]
19019 00001774 89C8                     mov     ax, cx
19020 00001776 83E03F                   and     ax, 3Fh         ; extract sector number
19021 00001779 39D8                     cmp     ax, bx          ; are we going to wrap?
19022 0000177B 764A                     jbe     short no_wrap
19023 0000177D F6F3                     div     bl              ; ah=new sector       #, al=# of head wraps
19024
19025                           ; we need to be careful here. if the new sector # is 0, then we are on the
19026                           ; last sector on that track.
19027
19028 0000177F 08E4                     or      ah, ah
19029 00001781 7503                     jnz     short not_on_bound
19030                                   ; 18/12/2022
19031 00001783 48                       dec     ax ; *
19032 00001784 88DC                     mov     ah, bl          ; set sector=BDS_BPB.BPB_SECTORSPERTRACK
19033                                                           ; if on boundary
19034                                   ;dec    al ; *          ; also decrement # of head wraps
19035                           not_on_bound:
19036 00001786 80E1C0                   and     cl, 0C0h        ; zero out sector #
19037 00001789 08E1                     or      cl, ah          ; or in new sector #
19038 0000178B 30E4                     xor     ah, ah          ; ax = # of head wraps
19039 0000178D 40                       inc     ax
19040 0000178E 00F0                     add     al, dh          ; add in starting head #
19041 00001790 80D400                   adc     ah, 0           ; catch any carry
19042                                   ; 02/09/2023
19043 00001793 268B5D15                 mov     bx, [es:di+15h]         ; [es:di+BDS.heads]
19044 00001797 39D8                     cmp     ax, bx
19045                                   ;cmp    ax, [es:di+15h]         ; [es:di+BDS.heads]
19046                                                           ; are we going to wrap around a      head?
19047 00001799 7632                     jbe     short no_wrap_head ; do      not lose new head number!!
19048 0000179B 52                       push    dx              ; preserve drive number and head number
19049 0000179C 31D2                     xor     dx, dx
19050                                   ;mov    bx, [es:di+15h]         ; [es:di+BDS.heads]
19051 0000179E F7F3                     div     bx              ; dx=new head #, ax=# of cylinder wraps
19052
19053                           ; careful here! if new head # is 0, then we are on the last head.
19054
19055 000017A0 09D2                     or      dx, dx
19056 000017A2 7507                     jnz     short no_head_bound
19057 000017A4 89DA                     mov     dx, bx          ; on boundary. set to BDS_BPB.BPB_HEADS
19058
19059                           ; if we had some cylinder wraps, we need to reduce them by one!!
19060
19061 000017A6 09C0                     or      ax, ax
19062 000017A8 7401                     jz      short no_head_bound
19063 000017AA 48                       dec     ax              ; reduce number       of cylinder wraps
19064                           no_head_bound:
19065 000017AB 88D7                     mov     bh, dl          ; bh has new head number
19066 000017AD 5A                       pop     dx              ; restore drive number and head number
19067 000017AE FECF                     dec     bh              ; get it 0-based
19068 000017B0 88FE                     mov     dh, bh          ; set up new head number in dh
19069 000017B2 88CF                     mov     bh, cl
19070 000017B4 80E73F                   and     bh, 3Fh         ; preserve sector number
19071 000017B7 B306                     mov     bl, 6
19072 000017B9 86D9                     xchg    cl, bl
19073 000017BB D2EB                     shr     bl, cl          ; get ms cylinder bits to ls end
19074 000017BD 00C5                     add     ch, al          ; add in cylinder wrap
19075 000017BF 10E3                     adc     bl, ah          ; add in high byte
19076 000017C1 D2E3                     shl     bl, cl          ; move up to ms       end
19077 000017C3 86CB                     xchg    bl, cl          ; restore cylinder bits      into cl
19078 000017C5 08F9                     or      cl, bh          ; or in sector number
19079                           no_wrap:
19080 000017C7 F8                       clc
19081 000017C8 5F                       pop     di
19082 000017C9 07                       pop     es
19083 000017CA 5B                       pop     bx
19084 000017CB 58                       pop     ax
19085 000017CC C3                       retn
19086                           ; ---------------------------------------------------------------------------
19087
19088                           no_wrap_head:
19089 000017CD 88C6                     mov     dh, al          ; do not lose new head number
19090 000017CF FECE                     dec     dh              ; get it 0-based
19091 000017D1 EBF4                     jmp     short no_wrap
19092
19093                           ; =============== S U B   R O U T I N E =======================================
```

```
19094
19095                                   ; 16/10/2022
19096
19097                                   ; this is a special version of the bds lookup code which is
19098                                   ; based on physical drives rather than the usual logical drives
19099                                   ; carry is set if the physical drive in dl is found, es:di -> its bds
19100                                   ; otherwise carry is clear
19101                                   ;
19102                                   ; guaranteed to trash no registers except es:di
19103
19104                                           ; 19/10/2022
19105                                   find_bds:
19106 000017D3 C43E[1901]                         les     di, [start_bds]       ; point es:di to first bds
19107                                   fbds_1:
19108 000017D7 26385504                           cmp     [es:di+4], dl  ; [es:di+BDS.drivenum]
19109 000017DB 7409                               jz      short fdbs_2
19110 000017DD 26C43D                             les     di, [es:di]    ; [es:di+BDS.link]
19111                                                                    ; go to next bds
19112 000017E0 83FFFF                             cmp     di, 0FFFFh
19113 000017E3 75F2                               jnz     short fbds_1
19114 000017E5 F9                                 stc
19115                                   fdbs_2:
19116 000017E6 C3                                 retn
19117
19118                                   ; =============== S U B   R O U TI N E =======================================
19119
19120                                   ; 16/10/2022
19121                                           ; 17/10/2022
19122                                   doint:
19123                                           ; 10/12/2022
19124 000017E7 8A5608                             mov     dl, [bp+8]     ; [bp+INT13FRAME.olddx]
19125                                                                    ; get physical drive number
19126                                           ; 19/10/2022 - Temporary !
19127                                           ;db     8Ah, 96h, 8, 0 ; mov dl, [bp+8]
19128
19129 000017EA 30E4                               xor     ah, ah
19130 000017EC 08C0                               or      al, al
19131 000017EE 7410                               jz      short dointdone       ; if zero sectors, return ax=0
19132                                           ; 10/12/2022
19133 000017F0 8A6603                             mov     ah, [bp+3]     ; [bp+INT13FRAME.oldax+1]
19134                                                                    ; get request code
19135                                           ;db     8Ah, 0A6h, 3, 0       ; mov ah, [bp+3]
19136 000017F3 FF7610                             push    word [bp+10h]  ; [bp+INT13FRAME.oldf]
19137                                           ;db     0FFh, 0B6h, 10h, 0 ; push word [bp+10h]
19138 000017F6 9D                                 popf
19139                                           ;call   far 70h:797h ; MSDOS 6.21 IO.SYS BIOSCODE:14EAh
19140                                           ; 17/10/2022
19141 000017F7 9A[0B07]7000                       call    DOSBIOSSEG:call_orig13
19142                                           ;;call  call_orig13    ; call far 70h:797h
19143                                                                    ; call far KERNEL_SEGMENT:call_orig13
19144 000017FC 9C                                 pushf
19145                                           ; 10/12/2022
19146 000017FD 8F4610                             pop     word [bp+10h]  ; [bp+INT13FRAME.oldf]
19147                                           ;db     8Fh, 86h, 10h, 0 ; pop word [bp+10h]
19148                                   dointdone:
19149 00001800 C3                                 retn
19150
19151                                   ;--------------------------------------------------------------------------
19152
19153                                   ; 16/10/2022
19154
19155                                   ; this is the true int 13 handler. we parse the request to see if there is
19156                                   ; a dma violation. if so, depending on the function, we:
19157                                   ;   read/write break the request into three pieces and move the middle one
19158                                   ;           into our internal buffer.
19159                                   ;
19160                                   ;   format     copy the format table into the buffer
19161                                   ;   verify     point the transfer address into the buffer
19162                                   ;
19163                                   ; this is the biggest bogosity of all. the ibm controller does not handle
19164                                   ; operations that cross physical 64k boundaries. in these cases, we copy
19165                                   ; the offending sector into the buffer below and do the i/o from there.
19166
19167                                   ;struc INT13FRAME
19168                                   ;.oldbp: resw
19169                                   ;.oldax: resw
19170                                   ;.oldbx: resw
19171                                   ;.oldcx: resw
19172                                   ;.olddx: resw
19173                                   ;.oldds: resw      ; now we save caller's ds, too
19174                                   ;.olddd: resd
19175                                   ;.oldf:    resw
19176                                   ;end struc
19177
19178                                   ;--------------------------------------------------------------------------
19179
19180                                   ;   entry conditions:
19181                                   ;   ah = function
19182                                   ;   al = number of sectors
19183                                   ;   es:bx = dma address
19184                                   ;   cx = packed track and sector
19185                                   ;   dx = head and drive
19186                                   ;   output conditions:
19187                                   ;   no dma violation.
19188
19189                                   ;   use extreme caution when working with this code. In general,
19190                                   ;     all registers are hot at all times.
19191                                   ;
19192                                   ;   question:  does this code handle cases where dma errors
19193                                   ;     occur during ecc retries, and where ecc errors occur during
19194                                   ;     dma breakdowns???? Hmmmmm.
19195
19196                                   ;--------------------------------------------------------------------------
19197
19198                                   ; ---------------------------------------------------------------------------
19199
19200                                           ; 26/12/2023 - Retro DOS v5.0
19201                                           ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:1889h
19202                                   dtype_array:
19203 00001801 90004000                           dd 400090h             ; 40h:90h is drive type array addr
19204
19205                                   ; 17/10/2022
19206                                   ;DTYPEARRAY equ dtype_array - DOSBIOSEG_2C7h ; (14F5h for MSDOS 5.0 IO.SYS)
19207                                   ; 09/12/2022
19208                                   DTYPEARRAY equ dtype_array
19209
19210                                   ; ---------------------------------------------------------------------------
19211
19212                                   ; stick some special stuff out of mainline
19213
19214                                   ; we know we're doing a format command. if we have changeline
19215                                   ; support, then flag some special changed stuff and set changed
19216                                   ; by format bit for all logical drives using this physical drive
19217
```

```
19218                                   format_special_stuff:
19219 00001805 803E[7700]00                       cmp     byte [fhave96], 0     ; do we have changeline support?
19220 0000180A 7459                               jz      short format_special_stuff_done ; brif not
19221 0000180C 53                                 push    bx
19222 0000180D BB4001                             mov     bx, 140h       ; fchanged_by_format+fchanged
19223 00001810 E85104                             call    set_changed_dl ; indicate that media changed by format
19224 00001813 5B                                 pop     bx
19225 00001814 EB4F                               jmp     short format_special_stuff_done
19226                                   ; ---------------------------------------------------------------------------
19227
19228                                   ; 16/10/2022
19229
19230                                   ; we know we've got ec35's on the system. Now see if we're doing
19231                                   ; a floppy. If so, create a mask and see if this particular
19232                                   ; drive is an ec35. If so, set dtype_array[drive]=93h
19233
19234                                             ; 19/10/2022
19235                                   ec35_special_stuff:
19236 00001816 84D2                               test    dl, dl         ; floppy or hard disk?
19237 00001818 7852                               js      short ec35_special_stuff_done ;      if hard drive, we're done
19238 0000181A 50                                 push    ax             ; see if this PARTICULAR drive is ec35
19239 0000181B 51                                 push    cx
19240 0000181C 88D1                               mov     cl, dl         ; turn drive number into bit map
19241 0000181E B001                               mov     al, 1          ; assume drive 0
19242 00001820 D2E0                               shl     al, cl         ; shift over correct number of times
19243 00001822 8406[A204]                         test    [ec35flag], al ; electrically compatible 3.5 incher?
19244 00001826 59                                 pop     cx
19245 00001827 58                                 pop     ax
19246 00001828 7442                               jz      short ec35_special_stuff_done
19247                                                                    ; done if this floppy is not an      ec35
19248 0000182A 53                                 push    bx             ; free up a far       pointer (es:bx)
19249 0000182B 06                                 push    es
19250                                             ; 17/10/2022
19251 0000182C 2EC41E[0118]                       les     bx, [cs:DTYPEARRAY]
19252                                             ;les    bx, dword ptr cs:DTYPEARRAY ; [cs:dtype_array]
19253                                                                    ; 0070h:3A65h =       2C7h:14F5h
19254 00001831 00D3                               add     bl, dl
19255 00001833 80D700                             adc     bh, 0          ; find entry for this drive
19256 00001836 26C60793                           mov     byte [es:bx], 93h ; establish drive type as:
19257                                                                    ; (360k disk in 360k drive,
19258                                                                    ; no double-stepping, 250 kbs transfer rate)
19259 0000183A 07                                 pop     es
19260 0000183B 5B                                 pop     bx
19261 0000183C EB2E                               jmp     short ec35_special_stuff_done
19262                                   ; ---------------------------------------------------------------------------
19263
19264                                   ; 16/10/2022
19265
19266                                   ; ps2_30 machine has some problem with ah=8h (read drive parm), int 13h.
19267                                   ; this function does not reset the common buses after the execution.
19268                                   ; to solve this problem, when we detect ah=8h, then we will save the result and
19269                                   ; will issue ah=1 (read status) call to reset the buses.
19270
19271                                   ps2_special_stuff:
19272 0000183E 803E[1E00]08                         cmp     byte [prevoper], 8 ; (ps2_30)
19273                                                                    ; read driver parm ?
19274 00001843 7407                               jz      short ps2_30_problem
19275 00001845 803E[1E00]15                       cmp     byte [prevoper], 15h
19276                                                                    ; apparently function 15h fails, too
19277 0000184A 752D                               jnz     short ps2_special_stuff_done
19278                                   ps2_30_problem:
19279 0000184C 50                                 push    ax
19280 0000184D B401                               mov     ah, 1
19281                                             ; 26/12/2023
19282                                             ;call   70h:70Bh ; PCDOS 7.1 IBMBIO.COM BIOSCODE:18D7h
19283                                             ;               ; call BIOSDATA:call_orig13
19284                                             ;;call  far 70:797h ; MSDOS 6.21 IO.SYS BIOSCODE:1543h
19285                                             ; 17/10/2022
19286 0000184F 9A[0B07]7000                       call    DOSBIOSSEG:call_orig13
19287                                             ;call   call_orig13    ; call far 70:797h
19288                                                                    ; call far KERNEL_SEGMENT:call_orig13
19289 00001854 58                                 pop     ax
19290 00001855 EB22                               jmp     short ps2_special_stuff_done
19291                                   ; ---------------------------------------------------------------------------
19292
19293                                   ; 17/10/2022
19294                                   ; 16/10/2022
19295
19296                                   ; here is the actual int13 handler
19297
19298                                   i13z:                                    ; 0070h:3ABBh =       02C7h:154Bh
19299
19300                                   ; cas -- inefficient! could push ds and load ds-> Bios_Data before
19301                                   ; vectoring up here from Bios_Data
19302
19303                                             ; 19/10/2022
19304 00001857 1E                                 push    ds             ; save caller's ds register first thing
19305                                             ;;mov   ds, word [cs:0030h]
19306                                                                    ; and set up our own ds -> Bios_Data
19307 00001858 2E8E1E[3000]                       mov     ds, [cs:BIOSDATAWORD]
19308                                             ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:0030h]
19309                                                                    ; = [02C7h:0030h] = [0070h:25A0h]
19310
19311                                   ; let the operation proceed. if there is a dma violation, then we do things
19312
19313 0000185D A3[1E00]                             mov     [prevoper], ax ; save request
19314 00001860 80FC05                             cmp     ah, 5          ; romformat
19315 00001863 74A0                               jz      short format_special_stuff
19316                                                                    ; go do special stuff for format
19317                                   format_special_stuff_done:
19318 00001865 803E[A204]00                       cmp     byte [ec35flag], 0 ; any electrically compat 3.5 inchers?
19319 0000186A 75AA                               jnz     short ec35_special_stuff
19320                                                                    ; go handle it out of line if so
19321                                   ec35_special_stuff_done:
19322                                             ; 26/12/2023
19323                                             ;call   70h:70Bh ; PCDOS 7.1 IBMBIO.COM BIOSCODE:18EDh
19324                                             ;;call  far 70:797h ; MSDOS 6.21 IO.SYS BIOSCODE:1560h
19325 0000186C 9A[0B07]7000                       call    DOSBIOSSEG:call_orig13
19326                                             ;call   call_orig13    ; call far KERNEL_SEGMENT:call_orig13
19327
19328 00001871 9C                                 pushf                  ; save result flags
19329
19330 00001872 803E[AF05]FA                       cmp     byte [model_byte], 0FAh ; is this a ps2/30?
19331                                                                    ; mdl_ps2_30
19332 00001877 74C5                               jz      short ps2_special_stuff
19333                                                                    ; exit mainline to address special
19334                                   ps2_special_stuff_done:                  ; ps2/30 problem if so
19335 00001879 9D                                 popf
19336 0000187A 7221                               jb      short goterr13 ; error on original orig13 call-thru?
19337                                   ret_from_i13:
19338 0000187C 1F                                 pop     ds
19339 0000187D CA0200                             retf    2              ; restore ds & iret w/flags
19340                                   ; ---------------------------------------------------------------------------
19341
```

```
                                ; most of our code exits through here. If carry isn't set, then
                                ; just do a simple exit. Else doublecheck that we aren't getting
                                ; a changeline error.

                                i13ret_ck_chglinerr:
00001880 73FA                           jnb     short ret_from_i13 ; done if not an error termination
                                i13_ret_error:
00001882 80FC06                         cmp     ah, 6           ; did i see a change event?
00001885 7513                           jnz     short int13b    ; skip if wrong       error
00001887 08D2                           or      dl, dl          ; is this for the hard disk?
00001889 780F                           js      short int13b    ; yes, ignore
0000188B 803E[7700]00                   cmp     byte [fhave96], 0
00001890 7408                           jz      short int13b    ; just in case ROM returned this
                                                                ; error even though it told us it
                                                                ; never would
00001892 53                             push    bx
00001893 BB4000                         mov     bx, 40h         ; fchanged
00001896 E8CB03                         call    set_changed_dl
00001899 5B                             pop     bx
                                int13b:
0000189A F9                             stc                     ; now return the error
0000189B EBDF                           jmp     short ret_from_i13
                                ; ---------------------------------------------------------------------------

                                ; some kind of error occurred. see if it is dma violation

                                goterr13:
0000189D 80FC09                         cmp     ah, 9           ; dma error?
000018A0 747C                           jz      short gotdmaerr
                                goterr13_xxxx:
000018A2 80FC11                         cmp     ah, 11h         ; ecc error?
000018A5 75DB                           jnz     short i13_ret_error ; other error. just     return back.
000018A7 803E[A905]01                   cmp     byte [media_set_for_format], 1 ; formatting?
000018AC 74D4                           jz      short i13_ret_error

000018AE 803E[1F00]02                   cmp     byte [prevoper+1], 2
                                        ;cmp    byte ptr ds:prevoper+1,      2 ; ecc-corrected error
                                                                ; (2 = romread)
                                                                ; ECC correction only applies to reads
000018B3 75CD                           jnz     short i13_ret_error

000018B5 30E4                           xor     ah, ah
                                        ;;call  far 70:797h ; MSDOS 6.21 IO.SYS BIOSCODE:15ABh
                                        ; 17/10/2022
000018B7 9A[0B07]7000                   call    DOSBIOSSEG:call_orig13
                                        ;call   call_orig13     ; call far KERNEL_SEGMENT:call_orig13
                                                                ; call far 70:797h
000018BC A1[1E00]                       mov     ax, [prevoper]
000018BF 30E4                           xor     ah, ah          ; return code =       no error
000018C1 3C01                           cmp     al, 1           ; if request for one sector, assume ok
000018C3 74B7                           jz      short ret_from_i13 ; return with carry clear
000018C5 53                             push    bx
000018C6 51                             push    cx
000018C7 52                             push    dx
000018C8 A2[2000]                       mov     [number_of_sec], al
                                loop_ecc:
000018CB B80102                         mov     ax, 201h        ; read one sector

                                ; we do reads one sector at a time. this ensures that we will eventually
                                ; finish the request since ecc errors on one sector do read in that sector.
                                ;
                                ; we need to put in some "intelligence" into the ecc handler to handle reads
                                ; that attempt to read more sectors than are available on a particular
                                ; track.
                                ;
                                ; we call check_wrap to set up the sector #, head # and cylinder # for
                                ; this request.
                                ;
                                ; at this point, all registers are set up for the call to orig13, except
                                ; that there may be a starting sector number that is bigger than the number
                                ; of sectors on a track.
                                ;
000018CE E88FFE                         call    check_wrap      ; get correct parameters for int 13
                                        ;;call  far 70:797h ; MSDOS 6.21 IO.SYS BIOSCODE:15C5h
                                        ; 17/10/2022
000018D1 9A[0B07]7000                   call    DOSBIOSSEG:call_orig13
                                        ;call   call_orig13     ; call far KERNEL_SEGMENT:call_orig13
000018D6 730C                           jnb     short ok11_op
000018D8 80FC09                         cmp     ah, 9           ; DMA error during ECC read?
000018DB 741B                           jz      short handle_dma_during_ecc
000018DD 80FC11                         cmp     ah, 11h         ; only allow ecc errors
000018E0 7510                           jnz     short ok11_exit_err
                                        ; 10/12/2022
                                        ; xor ax ax -> ah = 0
                                        ;mov    ah, 0           ; ecc error. reset the system again.
000018E2 31C0                           xor     ax, ax          ; clear the error code so that if this
                                                                ; was the last sector, no error      code
                                                                ; will be returned for the corrected
                                                                ; read. (clear carry too.)
                                ok11_op:
000018E4 FE0E[2000]                     dec     byte [number_of_sec]
000018E8 7409                           jz      short ok11_exit       ; all done?
000018EA FEC1                           inc     cl              ; advance sector number
                                                                ; add 200h to address
000018EC FEC7                           inc     bh
000018EE FEC7                           inc     bh
000018F0 EBD9                           jmp     short loop_ecc
                                ; ---------------------------------------------------------------------------

                                ; locate error returns centrally

                                ok11_exit_err:
000018F2 F9                             stc                     ; set carry bit again.
                                ok11_exit:
000018F3 5A                             pop     dx
000018F4 59                             pop     cx
000018F5 5B                             pop     bx
000018F6 EB88                           jmp     short i13ret_ck_chglinerr
                                ; ---------------------------------------------------------------------------

                                ; do the single sector read again, this time into our temporary
                                ; buffer, which is guaranteed not to have a DMA error, then
                                ; move the data to its proper location and proceed

                                handle_dma_during_ecc:
000018F8 06                             push    es
000018F9 53                             push    bx
000018FA BB[5201]                       mov     bx, disksector
000018FD 1E                             push    ds
000018FE 07                             pop     es              ; point es:bx to buffer
000018FF B80102                         mov     ax, 201h        ; read one sector
                                        ;;call  far 70:797h ; MSDOS 6.21 IO.SYS BIOSCODE:15F8h
                                        ; 17/10/2022
00001902 9A[0B07]7000                   call    DOSBIOSSEG:call_orig13
```

```
19466                                   ;call   call_orig13    ; call far KERNEL_SEGMENT:call_orig13
19467 00001907 5B                       pop     bx
19468 00001908 07                       pop     es
19469 00001909 7305                     jnb     short handle_dma_during_ecc_noerr
19470 0000190B 80FC11                   cmp     ah, 11h
19471 0000190E 75E2                     jnz     short ok11_exit_err ; if anything but ecc error, bomb out
19472
19473                                 ; now we're kosher. Copy the data to where it belongs and resume
19474                                 ; the ECC looping code.
19475
19476                                 handle_dma_during_ecc_noerr:
19477 00001910 56                       push    si
19478 00001911 57                       push    di
19479 00001912 89DF                     mov     di, bx
19480 00001914 BE[5201]                 mov     si, disksector
19481 00001917 E82BFE                   call    move_sector
19482 0000191A 5F                       pop     di
19483 0000191B 5E                       pop     si
19484 0000191C EBC6                     jmp     short ok11_op
19485                                 ; ---------------------------------------------------------------------------
19486
19487                                 ; we truly have a dma violation. restore register ax and retry the
19488                                 ; operation as best we can.
19489
19490                                 gotdmaerr:
19491 0000191E A1[1E00]                 mov     ax, [prevoper] ; 19/10/2022
19492 00001921 FB                       sti
19493 00001922 80FC02                   cmp     ah, 2           ; romread
19494 00001925 723B                     jb      short i13_done_dmaerr
19495                                                           ; just pass dma error thru for
19496                                                           ; functions we don't handle
19497 00001927 80FC04                   cmp     ah, 4           ; romverify
19498 0000192A 743C                     jz      short intverify
19499 0000192C 80FC05                   cmp     ah, 5           ; romformat
19500 0000192F 7448                     jz      short intformat
19501 00001931 772F                     ja      short i13_done_dmaerr
19502
19503                                 ; we are doing a read/write call. check for dma problems
19504
19505                                 ;   ******** set up stack frame here!!! ********
19506
19507 00001933 52                        push    dx
19508 00001934 51                       push    cx
19509 00001935 53                       push    bx
19510 00001936 50                       push    ax
19511 00001937 55                       push    bp
19512 00001938 89E5                     mov     bp, sp
19513 0000193A 8CC2                     mov     dx, es          ; check for 64k boundary error
19514                                   ; 26/12/2023
19515                                   ;add    dx, dx
19516                                   ;add    dx, dx
19517                                   ;add    dx, dx
19518                                   ;add    dx, dx          ; dx = dx*16
19519 0000193C D1E2                     shl     dx, 1
19520 0000193E D1E2                     shl     dx, 1
19521 00001940 D1E2                     shl     dx, 1
19522 00001942 D1E2                     shl     dx, 1           ; segment converted to absolute      address
19523 00001944 01DA                     add     dx, bx          ; combine with offset
19524 00001946 81C2FF01                 add     dx, 511         ; simulate a transfer
19525
19526                                 ; if carry is set, then we are within 512 bytes of the end of the segment.
19527                                 ; we skip the first transfer and perform the remaining buffering and transfer
19528
19529 0000194A 7303                      jnb     short no_skip_first
19530 0000194C E98300                   jmp     bufferx         ; restore dh=head & do buffer
19531                                 ; ---------------------------------------------------------------------------
19532
19533                                 no_skip_first:
19534 0000194F D0EE                     shr     dh, 1           ; dh = number of sectors before      address
19535 00001951 B480                     mov     ah, 128         ; ah = max number of sectors in      segment
19536 00001953 28F4                     sub     ah, dh
19537
19538                                 ; ah is now the number of sectors that we can successfully write in this
19539                                 ; segment. if this number is above or equal to the requested number, then we
19540                                 ; continue the operation as normal. otherwise, we break it into pieces.
19541                                 ;
19542                                 ; wait a sec. this is goofy. the whole reason we got here in the
19543                                 ; first place is because we got a dma error. so it's impossible
19544                                 ; for the whole block to fit, unless the dma error was returned
19545                                 ; in error.
19546
19547 00001955 38C4                       cmp     ah, al          ; can we fit it       in?
19548 00001957 7236                     jb      short doblock   ; no, perform blocking.
19549
19550                                 ; yes, the request fits. let it happen.
19551
19552 00001959 8A7609                    mov     dh, [bp+9]      ; [bp+INT13FRAME.olddx+1]
19553                                                             ; set up head number
19554 0000195C E888FE                   call    doint
19555 0000195F E9D900                   jmp     bad13           ; and return from this place
19556                                 ; ---------------------------------------------------------------------------
19557
19558                                 i13_done_dmaerr:
19559 00001962 B409                     mov     ah, 9           ; pass dma error thru to caller
19560 00001964 F9                       stc
19561 00001965 E914FF                   jmp     ret_from_i13    ; return with error,
19562                                                             ; we know it's not a changeline error
19563                                 ; ---------------------------------------------------------------------------
19564
19565                                 ; verify the given sectors. place the buffer pointer into our space.
19566
19567                                 intverify:
19568 00001968 06                       push    es              ; save caller's dma address
19569 00001969 53                       push    bx
19570 0000196A 1E                       push    ds              ; es:bx -> Bios_Data:disksector
19571 0000196B 07                       pop     es
19572                                 dosimple:
19573 0000196C BB[5201]                 mov     bx, disksector
19574                                                           ; do the i/o from Bios_Data:disksector
19575                                   ;;call far 70:797h ; MSDOS 6.21 IO.SYS BIOSCODE:1665h
19576                                   ; 17/10/2022
19577 0000196F 9A[0B07]7000             call    DOSBIOSSEG:call_orig13
19578                                   ;call   call_orig13    ; call far KERNEL_SEGMENT:call_orig13
19579 00001974 5B                       pop     bx
19580 00001975 07                       pop     es
19581 00001976 E907FF                   jmp     i13ret_ck_chglinerr
19582                                 ; ---------------------------------------------------------------------------
19583
19584                                 ; format operation. copy the parameter table into Bios_Data:disksector
19585
19586                                 intformat:
19587 00001979 06                       push    es
19588 0000197A 53                       push    bx
19589 0000197B 56                       push    si
```

```
19590 0000197C 57                              push    di
19591 0000197D 1E                              push    ds
19592
19593                                  ; point ds to the caller's dma buffer, es to Bios_Data
19594                                  ; in other words, swap (ds, es)
19595
19596 0000197E 06                              push    es
19597 0000197F 1E                              push    ds
19598 00001980 07                              pop     es
19599 00001981 1F                              pop     ds
19600 00001982 89DE                            mov     si, bx
19601 00001984 BF[5201]                        mov     di, disksector
19602 00001987 E8BBFD                          call    move_sector    ; user's data into Bios_Data:disksector
19603 0000198A 1F                              pop     ds
19604 0000198B 5F                              pop     di
19605 0000198C 5E                              pop     si             ; do the i/o from
19606 0000198D EBDD                            jmp     short dosimple ; Bios_Data:disksector
19607                                  ; ---------------------------------------------------------------------------
19608
19609                                  ; we can't fit the request into the entire block. perform the operation on
19610                                  ; the first block.
19611                                  ;
19612                                  ; doblock is modified to correctly handle multi-sector disk i/o.
19613                                  ; old doblock had added the number of sectors i/oed (ah in old doblock) after
19614                                  ; the doint call to cl. observing only the lower 6 bits of cl(=max. 64) can
19615                                  ; represent a starting sector, if ah was big, then cl would be clobbered.
19616                                  ; by the way, we still are going to use cl for this purpose since checkwrap
19617                                  ; routine will use it as an input. to prevent cl from being clobbered, a
19618                                  ; safe number of sectors should be calculated like "63 - # of sectors/track".
19619                                  ; doblock will handle the first block of requested sectors within the
19620                                  ; boundary of this safe value.
19621
19622                                          ; 26/12/2023 - Retro DOS v5.0
19623                                  doblock:
19624
19625                                  ; try to get the # of sectors/track from bds via rom drive number.
19626                                  ; for any mini disks installed, here we have to pray that they have the
19627                                  ; same # of sector/track as the main dos partition disk drive.
19628
19629 0000198F 8B5608                        mov     dx, [bp+8]     ; [bp+INT13FRAME.olddx]
19630                                                                 ; get head #, drive #
19631 00001992 51                              push    cx
19632 00001993 06                              push    es
19633 00001994 57                              push    di             ; ah - # of sectors before dma boundary
19634                                                                 ; al - requested # of sectors for i/o.
19635 00001995 E83BFE                          call    find_bds
19636 00001998 268B4D13                        mov     cx, [es:di+13h]       ; [es:di+BDS.secpertrack]
19637                                          ; 26/12/2023
19638 0000199C 26F6453F01                      test    byte [es:di+3Fh], 1
19639                                          ; 12/12/2022
19640                                          ;test  byte [es:di+23h], 1
19641                                          ;;test word [es:di+23h], 1 ; [es:di+BDS.flags],fnon_removable
19642 000019A1 5F                              pop     di
19643 000019A2 07                              pop     es
19644 000019A3 88E0                            mov     al, ah         ; set al=ah for       floppies
19645 000019A5 7404                            jz      short doblockflop ; they are track by track operation
19646 000019A7 B43F                            mov     ah, 63         ; ah = 63-secpt       (# safe sectors??)
19647 000019A9 28CC                            sub     ah, cl         ; al - # of sectors before dma boundary
19648                                  doblockflop:
19649 000019AB 59                              pop     cx
19650                                  doblockcontinue:
19651 000019AC 38C4                            cmp     ah, al         ; if safe_# >= #_of_sectors_to_go_before dma,
19652 000019AE 7305                            jnb     short doblocklast ; then #_of_sectors_to_go as it is for doint.
19653 000019B0 50                              push    ax
19654 000019B1 88E0                            mov     al, ah         ; otherwise, set al to ah to operate.
19655 000019B3 EB03                            jmp     short doblockdoint
19656                                  ; ---------------------------------------------------------------------------
19657
19658                                  doblocklast:
19659 000019B5 88C4                            mov     ah, al
19660 000019B7 50                              push    ax
19661                                  doblockdoint:                          ; let ah = al =       # of sectors for this shot
19662 000019B8 E82CFE                          call    doint
19663 000019BB 727E                            jb      short bad13    ; something happened, bye!
19664 000019BD 58                              pop     ax
19665 000019BE 286602                          sub     [bp+2], ah     ; sub [bp+INT13FRAME.oldax], ah
19666                                                                 ; decrement by the successful operation
19667 000019C1 00E1                            add     cl, ah         ; advance sector #. safety gauranteed.
19668 000019C3 00E7                            add     bh, ah         ; advance dma addres
19669 000019C5 00E7                            add     bh, ah         ; twice for 512 byte sectors
19670 000019C7 38C4                            cmp     ah, al         ; check the previous value
19671 000019C9 740A                            jz      short buffer   ; if #_of_sectors_to_go       < safe_#,
19672                                                                 ; then we are done already.
19673 000019CB 28E0                            sub     al, ah         ; otherwise,
19674                                                                 ; #_sector_to_go = #_of_sector_to_go - safe_#
19675 000019CD E890FD                          call    check_wrap     ; get new cx, dh for the next operation.
19676 000019D0 EBDA                            jmp     short doblockcontinue ;      handles next sectors left.
19677                                  ; ---------------------------------------------------------------------------
19678
19679                                  bufferx:
19680 000019D2 8A7609                          mov     dh, [bp+9]     ; [bp+INT13FRAME.olddx+1]
19681                                                                 ; set up head number
19682                                  buffer:
19683 000019D5 53                              push    bx
19684 000019D6 8A6603                          mov     ah, [bp+3]     ; [bp+INT13FRAME.oldax+1]
19685 000019D9 80FC03                          cmp     ah, 3          ; romwrite
19686 000019DC 7525                            jnz     short doread   ;
19687
19688                                  ; copy the offending sector into local buffer
19689
19690 000019DE 06                               push    es
19691 000019DF 1E                              push    ds
19692 000019E0 56                              push    si
19693 000019E1 57                              push    di
19694 000019E2 1E                              push    ds             ; exchange segment registers
19695 000019E3 06                              push    es
19696 000019E4 1F                              pop     ds
19697 000019E5 07                              pop     es
19698 000019E6 BF[5201]                        mov     di, disksector ; where to move
19699 000019E9 57                              push    di             ; save it
19700 000019EA 89DE                            mov     si, bx         ; source
19701 000019EC E856FD                          call    move_sector    ; move sector into local buffer
19702 000019EF 5B                              pop     bx             ; new transfer address
19703                                                                 ; (es:bx = Bios_Data:diskbuffer)
19704 000019F0 5F                              pop     di             ; restore caller's di & si
19705 000019F1 5E                              pop     si
19706 000019F2 1F                              pop     ds             ; restore Bios_Data
19707
19708                                  ; see if we are wrapping around a track or head
19709
19710 000019F3 B001                             mov     al, 1          ; [bp+INT13FRAME.olddx]
19711                                                                 ; get drive number
19712 000019F5 8A5608                          mov     dl, [bp+8]
19713 000019F8 E865FD                          call    check_wrap     ; sets up registers if wrap-around
```

```
                                                        ;
                                                        ; ah is function
                                                        ; al is 1 for single sector transfer
                                                        ; es:bx is local transfer addres
                                                        ; cx is track/sector number
                                                        ; dx is head/drive number
                                                        ; si,di unchanged
00001FFB E8E9FD                         call    doint
00001FFE 07                             pop     es              ; restore caller's dma segment
00001FFF 723A                           jb      short bad13     ; go clean up
00001A01 EB22                           jmp     short dotail
                                ; ---------------------------------------------------------------------------

                                ; reading a sector. do int first, then move things around

                                doread:
00001A03 06                             push    es
00001A04 53                             push    bx
00001A05 1E                             push    ds              ; es = Bios_Code
00001A06 07                             pop     es
00001A07 BB[5201]                       mov     bx, disksector
00001A0A B001                           mov     al, 1
00001A0C 8A5608                         mov     dl, [bp+8]      ; [bp+INT13FRAME.olddx]
                                                                ; get drive number
00001A0F E84EFD                         call    check_wrap      ;
                                                                ; ah = function
                                                                ; al = 1 for single sector
                                                                ; es:bx points to local       buffer
                                                                ; cx, dx are track/sector, head/drive
00001A12 E8D2FD                         call    doint
00001A15 5B                             pop     bx
00001A16 07                             pop     es
00001A17 7222                           jb      short bad13
00001A19 56                             push    si
00001A1A 57                             push    di
00001A1B 89DF                           mov     di, bx
00001A1D BE[5201]                       mov     si, disksector
00001A20 E822FD                         call    move_sector
00001A23 5F                             pop     di
00001A24 5E                             pop     si

                                ; note the fact that we've done 1 more sector

                                dotail:
00001A25 5B                             pop     bx              ; retrieve new dma area
00001A26 80C702                         add     bh, 2           ; advance over sector
00001A29 41                             inc     cx
00001A2A 8A4602                         mov     al, [bp+2]      ; [bp+INT13FRAME.oldax]
00001A2D F8                             clc
00001A2E FEC8                           dec     al
00001A30 7409                           jz      short bad13     ; no more i/o

                                ; see if we wrap around a track or head boundary with starting sector
                                ; we already have the correct head number to pass to check_wrap

00001A32 8A5608                         mov     dl, [bp+8]      ; [bp+INT13FRAME.olddx]
00001A35 E828FD                         call    check_wrap
00001A38 E8ACFD                         call    doint

                                ; we are done. ax has the final code; we throw away what we got before

                                ; M046  -- okay gang. Now we've either terminated our DMA loop,
                                ;       or we've finished. If carry is set now, our only
                                ;       hope for salvation is that it was a read operation
                                ;       and the error code is ECC error. In that case, we'll
                                ;       just pop the registers and go do the old ECC thing.
                                ;       When the DMA error that got us here in the first
                                ;       place occurs, it'll handle it.

                                bad13:
00001A3B 89EC                           mov     sp, bp
00001A3D 5D                             pop     bp
00001A3E 5B                             pop     bx
00001A3F 5B                             pop     bx
00001A40 59                             pop     cx
00001A41 5A                             pop     dx
00001A42 7203                           jb      short xgoterr13_xxxx ; go handle ECC errors
00001A44 E935FE                         jmp     ret_from_i13   ; non-error exit
                                ; ---------------------------------------------------------------------------

                                xgoterr13_xxxx:
00001A47 E958FE                         jmp     goterr13_xxxx

                                ; ---------------------------------------------------------------------------
                                        ; 10/12/2022
                                        ;db     0
                                ; ---------------------------------------------------------------------------

                                ;Bios_Code ends

                                ; 16/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)

                                ;--------------------------------------------------------------------------
                                ; MSBIO2.ASM - MSDOS 6.0 - 1991
                                ;--------------------------------------------------------------------------
                                ; 17/03/2019 - Retro DOS v4.0

                                        ; 19/10/2022
                                dsk_init:                       ; 2C7h:1742h = 70h:3CB2h
00001A4A 8A26[7500]                     mov     ah, [drvmax]
00001A4E BF[3C05]                       mov     di, dskdrvs
00001A51 1E                             push    ds              ; pass result in es:di
00001A52 07                             pop     es
00001A53 E934EC                         jmp     SetPtrSav

                                ; =============== S U B   R O U T I N E =======================================

                                ;--------------------------------------------------------------------------
                                ; install_bds installs a bds at location es:di into the current linked list of
                                ; bds maintained by this device driver. it places the bds at the end of the
                                ; list. Trashes (at least) ax, bx, di, si
                                ;--------------------------------------------------------------------------

                                        ; 26/12/2023 - Retro DOS v5.0
                                        ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:1AE0h
                                install_bds:
00001A56 1E                             push    ds              ; save Bios_Data segment
00001A57 BE[1901]                       mov     si, start_bds   ; beginning of chain

                                        ; ds:si now points to link to first bds
                                        ; assume bds list is non-empty
                                loop_next_bds:
00001A5A C534                           lds     si, [si]        ; [si+BDS.link]
                                                                ; fetch next bds
```

```
19838 00001A5C 268A4504                         mov     al, [es:di+4]  ; [es:di+BDS.drivenum]
19839 00001A60 384404                           cmp     [si+4],al      ; does this one       share a physical
19840                                                                  ; drive with new one?
19841 00001A63 7518                             jnz     short next_bds
19842 00001A65 B310                             mov     bl, 10h        ; fi_am_mult
19843                                           ; 26/12/2023
19844 00001A67 26085D3F                         or      [es:di+3Fh], bl
19845                                           ;or     [es:di+23h], bl       ; [es:di+BDS.flags]
19846                                                                  ; set both of them to i_am_mult      if so
19847 00001A6B 085C3F                           or      [si+3Fh], bl
19848                                           ;or     [si+23h], bl   ; [si+BDS.flags]
19849 00001A6E 2680653FDF                       and     byte [es:di+3Fh], 0DFh
19850                                           ;and    byte [es:di+23h], 0DFh ; [es:di+BDS.flags],~fi_own_physical
19851                                                                  ; we don't own it
19852 00001A73 8A5C3F                           mov     bl, [si+3Fh]
19853                                           ;mov    bl, [si+23h]   ; [si+BDS.flags]
19854                                                                  ; determine if changeline available
19855 00001A76 80E302                           and     bl, 2          ; fchangeline
19856 00001A79 26085D3F                         or      [es:di+3Fh], bl
19857                                           ;or     [es:di+23h], bl       ; [es:di+BDS.flags]
19858                                   next_bds:
19859                                           ; 02/09/2023 (PCDOS 7.1)
19860 00001A7D B8FFFF                           mov     ax, 0FFFFh     ; -1
19861 00001A80 3904                             cmp     [si], ax       ; [si+BDS.link],-1
19862                                           ;cmp    word [si], 0FFFFh ; [si+BDS.link],-1
19863                                                                  ; are we at end       of list?
19864 00001A82 75D6                             jnz     short loop_next_bds
19865 00001A84 8C4402                           mov     [si+2], es     ; [si+BDS.link+2],es
19866                                                                  ; install bds
19867 00001A87 893C                             mov     [si], di
19868 00001A89 268905                           mov     [es:di], ax    ; [es:di+BDS.link],-1
19869                                           ;mov    word [es:di], 0FFFFh ; [es:di+BDS.link],-1
19870                                                                  ; set next pointer to null
19871 00001A8C 1F                               pop     ds
19872
19873                                   ; 01/07/2023 - Retro DOS v4.2 (MSDOS 6.21 IO.SYS - BIOSCODE:1785h)
19874                                   ; 16/10/2022 (MSDOS 6.0 Code)
19875
19876                                   ; **** If the new drive has a higher EOT value, we must alter the
19877                                   ;      'eot' variable appropriately.
19878
19879                                           ; 26/12/2023
19880 00001A8D 268A4550                         mov     al, [es:di+50h]       ; [es:di+BDS.rsecpertrack]
19881                                           ; 01/06/2019
19882                                           ;mov    al,[es:di+52]
19883                                           ; 22/07/2023
19884                                           ;mov    al,[es:di+BDS.rsecpertrack]
19885 00001A91 3A06[2C01]                       cmp     al,[eot]
19886 00001A95 7603                             jbe     short _eot_ok
19887 00001A97 A2[2C01]                         mov     [eot],al
19888                                   _eot_ok:
19889 00001A9A C3                               retn
19890
19891                                   ; ---------------------------------------------------------------------------
19892
19893                                   ; 17/10/2022
19894                                   ;DRVLET     equ drvlet - DOSBIOSEG_2C7h
19895                                   ;SNGMSG     equ sngmsg - DOSBIOSEG_2C7h
19896                                   ; 09/12/2022
19897                                   DRVLET equ drvlet
19898                                   SNGMSG equ sngmsg
19899
19900                                   ; 16/10/2022
19901
19902                                   ;--------------------------------------------------------------------------
19903                                   ;  ask to swap the disk in drive a:
19904                                   ;   es:di -> bds
19905                                   ;   ds -> Bios_Data
19906                                   ;--------------------------------------------------------------------------
19907
19908                                           ; 26/12/2023 - Retro DOS v5.0
19909
19910                                           ; 19/10/2022
19911 00001A9B F606[1208]01             swpdsk:         test   byte [IsWin386], 1
19912                                           ;test   ds:IsWin386, 1 ; Is win386 present?
19913 00001AA0 7405                             jz      short no_win386       ; no, skip SetFocus
19914
19915                                           ; set focus to the correct VM
19916                                           ;call   far ptr 70h:813h ; PCDOS 7.1 IBMBIO.COM BIOSCODE:1B2Ch
19917                                           ;;call  far 70h:8D1h   ; MSDOS 6.21 IO.SYS BIOSCODE:179Ah
19918                                           ; 17/10/2022
19919 00001AA2 9A[1308]7000                     call    DOSBIOSSEG:V86_Crit_SetFocus ; BIOSDATA:V86_Crit_SetFocus
19920                                           ;call   far ptr V86_Crit_SetFocus ; call far 70h:8D1h
19921                                                                  ; call far KERNEL_SEGMENT:V86_Crit_SetFocus
19922                                   no_win386:
19923 00001AA7 51                               push    cx
19924 00001AA8 52                               push    dx
19925 00001AA9 268A5505                         mov     dl, [es:di+5]  ; [es:di+BDS.drivelet]
19926                                                                  ; get the drive       letter
19927
19928                                   ; WARNING : next two instructions assume that if the new disk is for drive B
19929                                   ;           then existing dsk is drive A & vice versa
19930
19931 00001AAD 88D6                             mov     dh, dl
19932 00001AAF 80F601                           xor     dh, 1
19933 00001AB2 29C9                             sub     cx, cx         ; nobody has handled swap disk
19934 00001AB4 B8004A                           mov     ax, 4A00h      ; multMULT<<8)|multMULTSWPDSK
19935                                                                  ; broadcast code for swap disk
19936                                                                  ; Broadcast it
19937 00001AB7 CD2F                             int     2Fh
19938 00001AB9 41                               inc     cx             ; cx == -1 ?
19939 00001ABA 741E                             jz      short swpdsk9  ; somebody has handled it
19940
19941                                   ; using a different drive in a one drive system so request the user change disks
19942
19943 00001ABC 80C241                           add     dl, 'A'
19944                                           ; 17/10/2022
19945 00001ABF 2E8816[F91A]                     mov     [cs:DRVLET], dl       ; "A: and press any key when ready\r\n\n"
19946                                           ; 16/10/2022
19947                                           ;;mov   byte [cs:drvlet], dl
19948                                           ;mov    byte ptr cs:17E4h, dl ; [cs:drvlet]
19949                                                                  ; 0070h:3D54h =       2C7h:17E4h
19950 00001AC4 BE[DD1A]                         mov     si, SNGMSG     ; "\r\nInsert diskette for drive "
19951                                           ;mov    si, 17C8h      ; sngmsg
19952                                                                  ; 0070h:3D38h =       2C7h:17C8h
19953 00001AC7 53                               push    bx
19954 00001AC8 2E                               cs
19955 00001AC9 AC                               lodsb                  ; get the next character of the message
19956                                           ;lods   byte ptr cs:[si]
19957                                   wrmsg_loop:
19958 00001ACA CD29                             int     29h            ; DOS 2+ internal - FAST PUTCHAR
19959                                                                  ; AL = character to display
19960 00001ACC 2E                               cs
19961 00001ACD AC                               lodsb
```

```
19962                                           ;lods   byte ptr cs:[si] ; cs lodsb
19963                                                                   ; get the next character of the      message
19964 00001ACE 08C0                             or      al, al
19965 00001AD0 75F8                             jnz     short wrmsg_loop
19966 00001AD2 E833E7                           call    con_flush       ; flush out keyboard queue
19967                                                                   ; call rom-bios
19968 00001AD5 30E4                             xor     ah, ah
19969 00001AD7 CD16                             int     16h             ; KEYBOARD - READ CHAR FROM BUFFER, WAIT IF EMPTY
19970                                                                   ; Return: AH = scan code, AL = character
19971 00001AD9 5B                               pop     bx
19972                                   swpdsk9:
19973 00001ADA 5A                               pop     dx
19974 00001ADB 59                               pop     cx
19975 00001ADC C3                               retn
19976
19977                                   ; ---------------------------------------------------------------------------
19978
19979                                   ; 16/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
19980
19981                                   ;--------------------------------------------------------
19982                                   ; include msbio.cl2 (MSDOS 6.0, 1991)
19983                                   ;--------------------------------------------------------
19984                                   ; (MSDOS 6.21 IO.SYS BIOSCODE:17D5h)
19985                                   ;--------------------------------------------------------
19986                                   ; 17/03/2019 - Retro DOS v4.0
19987                                   ; 26/12/2023 - Retro DOS v5.0
19988
19989                                               ; MSDOS 5.0 IO.SYS offset 0070h:3D38h or 02C7h:17C8h
19990                                               ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:1B67h
19991 00001ADD 0D0A                     sngmsg:             db 0Dh,0Ah
19992 00001ADF 496E73657274206469-               db 'Insert diskette for drive '
19992 00001AE8 736B6574746520666F-
19992 00001AF1 7220647269766520
19993
19994                                               ; MSDOS 5.0 IO.SYS offset 0070h:3D54h or 02C7h:17E4h
19995                                               ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:1B83h
19996 00001AF9 413A20616E64207072-      drvlet:             db 'A: and press any key when ready',0Dh,0Ah
19996 00001B02 65737320616E79206B-
19996 00001B0B 6579207768656E2072-
19996 00001B14 656164790D0A
19997 00001B1A 0A00                                 db 0Ah,0
19998
19999                                   ; =============== S U B   R O U T I N E =======================================
20000
20001                                   ;---------------------------------------------------------------------------
20002                                   ; input : es:di points to current bds for drive.
20003                                   ; return : zero set if no open files
20004                                   ;      zero reset if open files
20005                                   ;---------------------------------------------------------------------------
20006
20007                                               ; 26/12/2023 - Retro DOS v5.0
20008                                   chkopcnt:
20009 00001B1C 26837D3C00                           cmp     word [es:di+3Ch], 0
20010                                               ;cmp    word [es:di+20h], 0 ; [es:di+BDS.opcnt]
20011 00001B21 C3                                   retn
20012
20013                                   ; =============== S U B   R O U T I N E =======================================
20014
20015                                   ;---------------------------------------------------------------------------
20016                                   ; at media check time, we need to really get down and check what the change is.
20017                                   ; this is guaranteed to be expensive.
20018                                   ;
20019                                   ;   es:di -> bds, ds -> Bios_Data
20020                                   ;---------------------------------------------------------------------------
20021
20022                                               ; 26/12/2023 - Retro DOS v5.0
20023                                               ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:1BA6h
20024                                   mediacheck:
20025 00001B22 E852EE                               call    checksingle     ; make sure correct disk is in place
20026 00001B25 31F6                                 xor     si, si
20027 00001B27 E86101                               call    haschange
20028 00001B2A 742F                                 jz      short mediaret
20029                                               ; 26/12/2023
20030                                               ;test   byte [es:di+3Fh], 40h ; [es:di+BDS.flags], fchanged ; 40h
20031 00001B2C E85001                               call    checkromchange
20032 00001B2F 752B                                 jnz     short mediadovolid
20033 00001B31 50                                   push    ax
20034 00001B32 52                                   push    dx
20035 00001B33 268A5504                             mov     dl, [es:di+4]  ; [es:di+BDS.drivenum]
20036                                                                       ; set logical drive number
20037 00001B37 B416                                 mov     ah, 16h
20038 00001B39 CD13                                 int     13h             ; DISK - FLOPPY       DISK - CHANGE OF DISK STATUS (AT,XT2,XT286,CONV,PS)
20039                                                                       ; DL = drive to       check
20040                                                                       ; Return: AH = disk change status
20041 00001B3B 5A                                   pop     dx
20042 00001B3C 58                                   pop     ax
20043 00001B3D 721D                                 jb      short mediadovolid
20044 00001B3F BE0100                               mov     si, 1           ; signal no change
20045
20046                                   ; there are some drives with changeline that "lose" the changeline indication
20047                                   ; if a different drive is accessed after the current one. in order to avoid
20048                                   ; missing a media change, we return an "i don't know" to dos if the changeline
20049                                   ; is not active and we are accessing a different drive from the last one.
20050                                   ; if we are accessing the same drive, then we can safely rely on the changeline
20051                                   ; status.
20052                                               ; 19/10/2022
20053 00001B42 8A1E[1E01]                           mov     bl, [tim_drv]  ; get last drive accessed
20054 00001B46 26385D04                             cmp     [es:di+4], bl  ; [es:di+BDS.drivenum]
20055                                                                       ; (If the last drive accessed is not current drive
20056                                                                       ; media change status may be incorrect.      So,
20057                                                                       ; "I don't now" will be returned even if it is indicated
20058                                                                       ; as media is not changed.)
20059 00001B4A 740F                                 jz      short mediaret ; (same drive,
20060                                                                       ; media changeline indication is reliable)
20061
20062                                   ; do the 2 second twiddle. if time >= 2 seconds, do a volid check.
20063                                   ; otherwise return "i don't know" (strictly speaking, we should return a
20064                                   ; "not changed" here since the 2 second test said no change.)
20065
20066 00001B4C 50                                   push    ax
20067 00001B4D 51                                   push    cx
20068 00001B4E 52                                   push    dx
20069 00001B4F E8D8EA                               call    Check_Time_Of_Access
20070 00001B52 5A                                   pop     dx
20071 00001B53 59                                   pop     cx
20072 00001B54 58                                   pop     ax
20073 00001B55 09F6                                 or      si, si
20074 00001B57 7403                                 jz      short mediadovolid ; check_time      says ">= 2 secs       passed"
20075                                                                       ; (volume id will be checked)
20076 00001B59 31F6                                 xor     si, si         ; return "i don't know"
20077                                   mediaret:
20078 00001B5B C3                                   retn
20079                                   ; ---------------------------------------------------------------------------
20080
```

```
                                ; somehow the media was changed. look at vid to see. we do not look at fat
                                ; because this may be different since we only set medbyt when doing a read
                                ; or write.

                                mediadovolid:
00001B5C E877EB                         call    GetBp           ; build a new bpb in current bds
00001B5F 72FA                           jb      short mediaret
00001B61 E82D00                         call    check_vid
00001B64 73F5                           jnb     short mediaret
00001B66 E940F2                         jmp     maperror        ; fix up al for       return to dos
                                ; ---------------------------------------------------------------------------

                                ; simple, quick check of latched change. if no indication, then return
                                ; otherwise do expensive check. if the expensive test fails, pop off the
                                ; return and set al = 15 (for invalid media change) which will be returned to
                                ; dos.
                                ;
                                ; for dos 3.3, this will work only for the drive that has changeline.

                                ;   call with es:di -> bds, ds -> Bios_Data
                                ;   ***** warning:  this routine will return one level up on the stack
                                ;                   if an error occurs!

                                checklatchio:

                                ; if returning fake bpb then assume the disk has not changed

                                        ; 26/12/2023
                                        ;cmp    word [es:di+3Ch], 0 ; [es:di+BDS.opcnt]
00001B69 E8B0FF                         call    chkopcnt
00001B6C 741B                           jz      short checkret ; done if zero

                                ; check for past rom indications. if no rom change indicated, then return ok.

                                        ; 26/12/2023
                                        ;test   word [es:di+3Fh], 40h
                                        ;                       ; test [es:di+BDS.flags], fchanged ; 40h
00001B6E E80E01                         call    checkromchange
00001B71 7416                           jz      short checkret

                                ; we now see that a change line has been seen in the past. let's do the
                                ; expensive verification.

00001B73 E860EB                         call    GetBp           ; build bpb in current bds
00001B76 720F                           jb      short ret_no_error_map ; getbp has already called maperror
00001B78 E81600                         call    check_vid
00001B7B 7207                           jb      short checklatchret ; disk error trying     to read in.
00001B7D 09F6                           or      si, si          ; is changed for sure?
00001B7F 7908                           jns     short checkret
00001B81 E88F00                         call    returnvid
                                checklatchret:
00001B84 E822F2                         call    maperror        ; fix up al for       return to dos
                                ret_no_error_map:
00001B87 F9                             stc
00001B88 5E                             pop     si              ; pop off return address
                                checkret:
00001B89 C3                             retn
                                ; ---------------------------------------------------------------------------

                                ; check the fat and the vid. return in di -1 or 0. return with carry set
                                ; only if there was a disk error. return that error code in ax.
                                ;
                                ;   called with es:di -> bds, ds -> Bios_Data

                                checkfatvid:
00001B8A E8D101                         call    fat_check       ; check the fat and the vid
00001B8D 09F6                           or      si, si
00001B8F 7835                           js      short changed_drv

                                ; the fat was the same. fall into check_vid and check volume id.

                                        ; fall into check_vid

                                ; =============== S U B   R O U T I N E =======================================

                                ; now with the extended boot record, the logic should be enhanced.
                                ;
                                ; if it is the extended boot record, then we check the volume serial
                                ; number instead of volume id. if it is different, then set si to -1.
                                ;
                                ; if it is same, then si= 1 (no change).
                                ;
                                ; if it is not the extended boot record, then just follows the old
                                ; logic. dos 4.00 will check if the # of fat in the boot record bpb
                                ; is not 0.  if it is 0 then it must be non_fat based system and
                                ; should have already covered by extended boot structure checking.
                                ; so, we will return "i don't know" by setting si to 0.
                                ;
                                ; this routine assume the newest valid boot record is in cs:[disksector].
                                ; (this will be gauranteed by a successful getbp call right before this
                                ; routine.)
                                ;
                                ;   called with es:di -> bds, ds -> bds

                                        ; 26/12/2023 - Retro DOS v5.0
                                        ; 19/10/2022
                                check_vid:

                                ; check the disksector.EXT_BOOT_SIG variable for the extended
                                ; boot signature. if it is set then go to do the extended
                                ; id check otherwise continue with code below

                                        ; 26/12/2023
                                        ;;;
00001B91 833E[6801]00                   cmp     word [disksector+16h], 0 ; BPB_FATSz16
00001B96 7507                           jnz      short chk_vid_1
00001B98 803E[9401]29                   cmp      byte [disksector+42h], 29h ; BS_FAT32_BootSig
                                                                ; [disksector+EXT_BOOT.SIG],EXT_BOOT_SIGNATURE
00001B9D EB05                           jmp      short chk_vid_2
                                chk_vid_1:
                                        ;;;
00001B9F 803E[7801]29                   cmp     byte [disksector+26h], 29h
                                                                ; [disksector+EXT_BOOT.SIG],
                                                                ; EXT_BOOT_SIGNATURE
                                chk_vid_2:      ; 26/12/2023
00001BA4 7427                           jz      short do_ext_check_id
00001BA6 E8E200                         call    haschange
00001BA9 74DE                           jz      short checkret
00001BAB 31F6                           xor     si, si
00001BAD 803E[6201]00                   cmp     byte [disksector+10h], 0 ; BPB_NumFATs
                                                                ; [disksector+EXT_BOOT.BPB+EBPB.NUMBEROFFATS]
00001BB2 7411                           jz      short checkfatret ; don't read vol id
                                                                ; if not fat system
00001BB4 E8F400                         call    read_volume_id
```

```
20205 00001BB7 720C                           jb      short checkfatret
20206 00001BB9 E89901                         call    check_volume_id
20207 00001BBC BEFFFF                         mov     si, 0FFFFh      ; -1
20208                                                                 ; definitely changed
20209 00001BBF 7505                           jnz     short changed_drv
20210
20211 00001BC1 46                             inc     si              ; not changed
20212                                 vid_no_changed:
20213 00001BC2 E8C000                         call    resetchanged
20214                                         ; 12/12/2022
20215                                         ; cf=0 ('and' instruction in 'resetchanged' clears cf)
20216                                         ;clc
20217                                 checkfatret:
20218 00001BC5 C3                             retn
20219                                 ; ---------------------------------------------------------------------------
20220
20221                                         ; 12/12/2022
20222                                 changed_drv:
20223 00001BC6 F8                             clc                     ; cas -- return       no error
20224 00001BC7 C606[1E01]FF                   mov     byte  [tim_drv], 0FFh
20225                                                                 ; ensure that we ask rom for media
20226 00001BCC C3                             retn                    ; check next time round
20227                                 ; ---------------------------------------------------------------------------
20228
20229                                 ; extended id check
20230
20231                                 ; 16/10/2022
20232
20233                                 ; the code to check extended id is basically a check to see if the
20234                                 ; volume serial number is still the same. the volume serial number
20235                                 ; previously read is in cs:disksector.EXT_BOOT_SERIAL
20236                                 ; ds:di points to the bds of the drive under consideration.
20237                                 ; the bds has fields containing the high and low words
20238                                 ; of the volume serial number of the media in the drive.
20239                                 ; compare these fields to the fields mentioned above. if these fields
20240                                 ; do not match the media has changed and so we should jump to the code
20241                                 ; starting at ext_changed else return "i don't know" status
20242                                 ; in the register used for the changeline status and continue executing
20243                                 ; the code given below. for temporary storage use the register which
20244                                 ; has been saved and restored around this block.
20245                                 ;
20246                                 ; bds fields in inc\msbds.inc
20247
20248                                         ; 26/12/2023 - Retro DOS v5.0
20249                                         ; 19/10/2022
20250                                 do_ext_check_id:
20251                                         ; 26/12/2023
20252                                         ;push   ax
20253                                         ;;mov   ax, word ptr ds:disksector+27h
20254                                         ;                       ; [DiskSector+EXT_BOOT.SERIAL]
20255                                         ;mov    ax, [disksector+27h]
20256                                 ; 26/12/2023
20257                                 %if 1
20258                                         ;;;
20259 00001BCD 57                             push    di
20260 00001BCE BE[9501]                       mov     si, disksector+43h ; BS_FAT32_VolID
20261                                                                 ; [DiskSector+FAT32_EXT_BOOT.SERIAL]
20262 00001BD1 833E[6801]00                   cmp     word [disksector+16h], 0 ; BPB_FATSz16
20263 00001BD6 7403                           jz      short chk_vid_3
20264 00001BD8 83EE1C                         sub     si, 28          ; BS_VolID
20265                                         ; si = disksector+27h ; [DiskSector+EXT_BOOT.SERIAL]
20266                                 chk_vid_3:
20267                                         ; [es:di+89h] = [es:di+BDS.vol_serial]
20268 00001BDB 81C78900                       add     di, 137         ; BDS.vol_serial
20269 00001BDF A7                             cmpsw   ; [DiskSector+EXT_BOOT.SERIAL] ; (or FAT32_EXT_BOOT)
20270                                                 ;               = [di+BDS.vol_serial] ?
20271 00001BE0 7501                           jnz     short chk_vid_4
20272 00001BE2 A7                             cmpsw   ; [DiskSector+EXT_BOOT.SERIAL+2] ; (or FAT32_EXT_BOOT)
20273                                                 ;               = [di+BDS.vol_serial+2] ?
20274                                 chk_vid_4:
20275 00001BE3 5F                             pop     di
20276                                         ;pop    ax
20277 00001BE4 7504                           jnz     short ext_changed ; not equal/same
20278 00001BE6 31F6                           xor     si, si          ; 0 ; don't know
20279 00001BE8 EBD8                           jmp     short vid_no_changed ; reset the flag
20280                                         ;;;
20281                                 %else
20282                                         ; 02/09/2023
20283                                         xor     si, si ; 0
20284                                         cmp     ax, [es:di+57h]       ; [di+BDS.vol_serial]
20285                                         jnz     short ext_changed
20286                                         mov     ax, [disksector+29h] ; [DiskSector+EXT_BOOT.SERIAL+2]
20287                                         cmp     ax, [es:di+59h]       ; [di+BDS.vol_serial+2]
20288                                         jnz     short ext_changed
20289                                         ;xor    si, si          ; 0
20290                                                                 ; don't know
20291                                         pop     ax
20292                                         jmp     short vid_no_changed
20293                                                                 ; reset the flag
20294                                 %endif
20295
20296                                 ; ---------------------------------------------------------------------------
20297
20298                                 ext_changed:
20299                                         ; 26/12/2023
20300                                         ;pop    ax
20301                                         ; 02/09/2023
20302                                         ;dec    si ; mov si, 0FFFFh ; -1
20303 00001BEA BEFFFF                         mov     si, 0FFFFh      ; -1
20304                                                                 ; disk changed!
20305                                         ; 12/12/2022
20306                                         ; ('changed_drv' clears cf)
20307                                         ;clc
20308 00001BED EBD7                           jmp     short changed_drv
20309
20310                                 ; ---------------------------------------------------------------------------
20311
20312                                 ; at i/o time, we detected the error. now we need to determine whether the
20313                                 ; media was truly changed or not. we return normally if media change unknown.
20314                                 ; and we pop off the call and jmp to harderr if we see an error.
20315                                 ;
20316                                 ; es:di -> bds
20317
20318                                 checkio:
20319 00001BEF 80FC06                         cmp     ah, 6
20320 00001BF2 75D1                           jnz     short checkfatret
20321 00001BF4 E825FF                         call    chkopcnt
20322 00001BF7 74CC                           jz      short checkfatret
20323 00001BF9 E8DAEA                         call    GetBp
20324 00001BFC 7212                           jb      short no_error_map
20325 00001BFE E889FF                         call    checkfatvid
20326 00001C01 7209                           jb      short checkioret ; disk       error trying to       read in.
20327 00001C03 09F6                           or      si, si          ; is changed for sure?
20328 00001C05 7802                           js      short checkioerr ; yes changed
```

```
20329 00001C07 45                          inc     bp              ; allow a retry
20330 00001C08 C3                          retn
20331                                  ; ---------------------------------------------------------------------------
20332
20333                                  checkioerr:
20334 00001C09 E80700                      call    returnvid
20335
20336                                  checkioret:
20337 00001C0C F9                          stc                     ; make sure carry gets passed through
20338 00001C0D E955F1                      jmp     harderr
20339                                  ; ---------------------------------------------------------------------------
20340
20341                                  no_error_map:
20342 00001C10 E955F1                      jmp     harderr2
20343
20344                                  ; =============== S U B   R O U TI N E =======================================
20345
20346                                  ; return vid sets up the vid for a return to dos.
20347                                  ;  es:di -> bds, returns pointer in packet to bds_volid
20348                                  ;  **** trashes si! ****
20349
20350                                  returnvid:
20351 00001C13 BE1600                      mov     si, 22          ; extra
20352                                                              ; offset into pointer to return      value
20353 00001C16 E80700                      call    vid_into_packet
20354 00001C19 B406                        mov     ah, 6
20355 00001C1B F9                          stc
20356 00001C1C C3                          retn
20357
20358                                  ; ---------------------------------------------------------------------------
20359
20360                                  ; moves the pointer to the volid for the drive into the original request packet
20361                                  ; no attempt is made to preserve registers.
20362                                  ;
20363                                  ; assumes es:di -> bds
20364                                  ; **trashes si**
20365
20366                                  media_set_vid:
20367 00001C1D BE0F00                      mov     si, 15          ; trans+1
20368                                                              ; return the value here      in packet
20369
20370                                      ; fall into vid_into_packet
20371
20372                                  ; =============== S U B   R O U TI N E =======================================
20373
20374                                  ; return pointer to vid in bds at es:di in packet[si]
20375
20376                                      ; 14/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
20377                                      ; 19/10/2022
20378                                  vid_into_packet:
20379 00001C20 1E                          push    ds              ; return pointer to vid      in bds at es:di       in packet[si]
20380 00001C21 C51E[1200]                  lds     bx, [ptrsav]
20381                                      ;add    di, 75          ; BDS.volid
20382                                      ; 14/04/2024
20383 00001C25 83C77D                      add     di, 125 ; (PCDOS 7.1)
20384 00001C28 8938                        mov     [bx+si], di
20385                                      ;sub    di, 75          ; BDS.volid
20386 00001C2A 83EF7D                      sub     di, 125
20387 00001C2D 8C4002                      mov     [bx+si+2], es
20388 00001C30 1F                          pop     ds
20389                                  dofloppy:   ; 18/12/2022
20390 00001C31 C3                          retn
20391
20392                                  ; ---------------------------------------------------------------------------
20393
20394                                  ;---------------------------------------------------------------------------
20395                                  ;   hidensity - examine a drive/media descriptor to set the media type. if
20396                                  ;   the media descriptor is not f9 (not 96tpi or 3 1/2), we return and let the
20397                                  ;   caller do the rest. otherwise, we pop off the return and jump to the tail
20398                                  ;   of getbp. for 3.5" media, we just return.
20399                                  ;
20400                                  ;   inputs: es:di point to correct bds for this drive
20401                                  ;           ah has media byte
20402                                  ;
20403                                  ;   outputs:       carry clear
20404                                  ;               no registers modified
20405                                  ;           carry set
20406                                  ;               al = sectors/fat
20407                                  ;               bh = number of root directory entries
20408                                  ;               bl = sectors per track
20409                                  ;               cx = number of sectors
20410                                  ;               dh = sectors per allocation unit
20411                                  ;               dl = number of heads
20412                                  ;
20413                                  ;---------------------------------------------------------------------------
20414
20415                                      ; 26/12/2023 - Retro DOS v5.0
20416                                  hidensity:
20417
20418                                  ; check for correct drive
20419
20420                                      ; 26/12/2023
20421 00001C32 26F6453F02                  test    byte [es:di+3Fh], 2 ; is it special?
20422                                      ; 12/12/2022
20423                                      ;test   byte [es:di+23h], 2
20424                                      ;;test  word [es:di+23h], 2 ; is it special?
20425                                                              ; [es:di+BDS.flags], fchangeline
20426 00001C37 74F8                        jz      short dofloppy ; no, do normal floppy test
20427
20428                                  ; we have a media byte that is pretty complex. examine drive information
20429                                  ; table to see what kind it is.
20430
20431                                      ; 26/12/2023
20432 00001C39 26807D3E02                  cmp     byte [es:di+3Eh], 2 ; is it single-media?
20433                                      ;cmp    byte [es:di+22h], 2 ; is it single-media?
20434 00001C3E 74F1                        jz      short dofloppy ; [es:di+BDS.formfactor], ffSmall
20435                                                              ; yes, use fatid...
20436                                  ; 96 tpi drive?
20437 00001C40 80FCF9                      cmp     ah, 0F9h
20438 00001C43 75EC                        jnz     short dofloppy
20439
20440                                  ;------ If formfactor of drive = ffother or ff288 it has to be
20441                                  ;------ a 720K diskette
20442
20443                                      ; 02/09/2023 (PCDOS 7.1)
20444                                      ; 26/12/2023
20445 00001C45 268A453E                    mov     al, [es:di+3Eh] ; [es:di+BDS.formfactor]
20446                                      ;mov    al, [es:di+22h]       ; [es:di+BDS.formfactor]
20447 00001C49 3C07                        cmp     al, 7
20448                                      ;cmp    byte [es:di+22h], 7 ; [es:di+BDS.formfactor]
20449                                                              ; ffOther
20450 00001C4B 7413                        jz      short Is720K
20451 00001C4D 3C09                        cmp     al, 9
20452                                      ;cmp    byte [es:di+22h], 9 ; [es:di+BDS.formfactor]
```

```
20453                                                   ; ff288
20454 00001C4F 740F                             jz      short Is720K
20455 00001C51 B007                             mov     al, 7           ; seven sectors / fat
20456 00001C53 BB0FE0                           mov     bx, 57359       ; 224*256+0Fh
20457                                                   ; 224 root dir entries
20458                                                   ; & 0Fh sector max
20459 00001C56 B96009                           mov     cx, 2400        ; 80*15*2
20460                                                   ; 80 tracks, 15 sectors/track,
20461                                                   ; 2 sides
20462                                           ; 02/09/2023
20463 00001C59 5A                               pop     dx              ; pop off return address
20464 00001C5A BA0201                           mov     dx, 258         ; 1*256+2
20465                                                   ; sectors/allocation unit
20466                                                   ; & head max
20467                                           ;add    sp, 2           ; pop off return address
20468 00001C5D E9EAEA                           jmp     Has1            ; return to tail of getbp
20469                                   ; ---------------------------------------------------------------------------
20470
20471                                   Is720K:
20472                                           ; 02/09/2023
20473 00001C60 5B                               pop     bx              ; pop off return address
20474                                           ;add    sp, 2           ; pop off return address
20475 00001C61 E9A9EA                           jmp     Has720K         ; return to 720K code
20476                                   ; ---------------------------------------------------------------------------
20477
20478                                           ; 18/12/2022
20479                                   ;dofloppy:
20480                                           ;retn
20481
20482                                   ; =============== S U B   R O U T I N E =======================================
20483
20484                                   ; 16/10/2022
20485
20486                                   ;---------------------------------------------------------------------------
20487                                   ; set_changed_dl - sets flag bits according to bits set in bx.
20488                                   ;             essentially used to indicate changeline, or format.
20489                                   ;
20490                                   ;   inputs: dl contains physical drive number
20491                                   ;           bx contains bits to set in the flag field in the bdss
20492                                   ;   outputs:       none
20493                                   ;   registers modified: flags
20494                                   ;
20495                                   ;   called from int13 hooker.  Must preserve ALL registers!!!
20496                                   ;
20497                                   ; in the virtual drive system we *must* flag the other drives as being changed
20498                                   ;---------------------------------------------------------------------------
20499
20500                                           ; 26/12/2023 - Retro DOS v5.0
20501                                   set_changed_dl:
20502 00001C64 06                               push    es
20503 00001C65 57                               push    di
20504                                           ;les    di, ds:start_bds
20505                                           ; 19/10/2022
20506 00001C66 C43E[1901]                       les     di, [start_bds]
20507
20508                                   ; note: we assume that the list is non-empty
20509
20510                                   scan_bds:
20511 00001C6A 26385504                         cmp     [es:di+4], dl  ; [es:di+BDS.drivenum]
20512 00001C6E 7504                             jnz     short get_next_bds
20513
20514                                   ; someone may complain, but this *always* must be done when a disk change is
20515                                   ; noted. there are *no* other compromising circumstances.
20516
20517                                           ; 26/12/2023
20518 00001C70 26095D3F                         or      [es:di+3Fh], bx       ; [es:di+BDS.flags]
20519                                           ;or     [es:di+23h], bx       ; [es:di+BDS.flags]
20520                                                                   ; signal change       on other drive
20521                                   get_next_bds:
20522 00001C74 26C43D                           les     di, [es:di]     ; [es:di+BDS.link]
20523                                                                   ; go to next bds
20524 00001C77 83FFFF                           cmp     di, 0FFFFh
20525 00001C7A 75EE                             jnz     short scan_bds ; loop unless we hit end of chain
20526 00001C7C 5F                               pop     di
20527 00001C7D 07                               pop     es
20528 00001C7E C3                               retn
20529
20530                                   ; =============== S U B   R O U T I N E =======================================
20531
20532                                   ;---------------------------------------------------------------------------
20533                                   ; checkromchange - see if external program has diddled rom change line.
20534                                   ;
20535                                   ;   inputs: es:di points to current bds.
20536                                   ;   outputs:       zero set - no change
20537                                   ;           zero reset - change
20538                                   ;   registers modified: none
20539                                   ;---------------------------------------------------------------------------
20540
20541                                           ; 26/12/2023 - Retro DOS v5.0
20542                                   checkromchange:
20543                                           ;test   word [es:di+BDS.flags], fchanged ; 40h
20544                                           ; 26/12/2023
20545 00001C7F 26F6453F40                       test    byte [es:di+3Fh], 40h
20546                                           ; 10/12/2022
20547                                           ;test   byte [es:di+23h], 40h
20548                                           ;;test  word [es:di+23h], 40h ; [es:di+BDS.flags]
20549                                                                   ; fchanged
20550 00001C84 C3                               retn
20551
20552                                   ; =============== S U B   R O U T I N E =======================================
20553
20554                                   ;---------------------------------------------------------------------------
20555                                   ; resetchanged - restore value of change line
20556                                   ;
20557                                   ;   inputs: es:di points to current bds
20558                                   ;   outputs:       none
20559                                   ;   registers modified: none
20560                                   ;---------------------------------------------------------------------------
20561
20562                                           ; 26/12/2023 - Retro DOS v5.0
20563                                   resetchanged:
20564                                           ;and    word [es:di+BDS.flags], ~fchanged ; 0FFBFh
20565                                           ; 26/12/2023
20566 00001C85 2680653FBF                       and     byte [es:di+3Fh], 0BFh
20567                                           ; 10/12/2022
20568                                           ;and    byte [es:di+23h], 0BFh
20569                                           ;;and   word [es:di+23h], 0FFBFh ; [es:di+BDS.flags]
20570                                                                   ; ~fchanged
20571 00001C8A C3                               retn
20572
20573                                   ; =============== S U B   R O U T I N E =======================================
20574
20575                                   ;---------------------------------------------------------------------------
20576                                   ; haschange - see if drive can supply change line
```

```
20577                                   ;
20578                                   ;   inputs: es:di points to current bds
20579                                   ;   outputs:       zero set - no change line available
20580                                   ;           zero reset - change line available
20581                                   ;   registers modified: none
20582                                   ;-----------------------------------------------------------------------
20583                                   
20584                                               ; 26/12/2023 - Retro DOS v5.0
20585                                   haschange:
20586                                               ;test  word [es:di+BDS.flags], fchangeline ; 2
20587                                               ; 26/12/2023
20588 00001C8B 26F6453F02                           test   byte [es:di+3Fh], 2
20589                                               ; 10/12/2022
20590                                               ;test  byte [es:di+23h], 2
20591                                               ;;test  word [es:di+23h], 2 ; [es:di+BDS.flags]
20592                                                                           ; fchangeline
20593 00001C90 C3                                   retn
20594                                   
20595                                   ; ---------------------------------------------------------------------------
20596                                   
20597                                   ; 16/10/2022
20598                                   
20599                                   ;-----------------------------------------------------------------------
20600                                   ; set_volume_id      -    main routine, calls other routines.
20601                                   ; read_volume_id     -    read the volume id and tells if it has been changed.
20602                                   ; transfer_volume_id -    copy the volume id from tmp to special drive.
20603                                   ; check_volume_id    -    compare volume id in tmp area with one expected for drive.
20604                                   ; fat_check          -    see of the fatid has changed in the specified drive.
20605                                   ;-----------------------------------------------------------------------
20606                                   
20607                                   ; set_volume_id
20608                                   ;   if drive has changeline support, read in and set the volume_id
20609                                   ; and the last fat_id byte. if no change line support then do nothing.
20610                                   ;
20611                                   ;   on entry:
20612                                   ;   es:di points to the bds for this disk.
20613                                   ;   ah contains media byte
20614                                   ;
20615                                   ;   on exit:
20616                                   ;   carry clear:
20617                                   ;      successful call
20618                                   ;   carry set
20619                                   ;      error and ax has error code
20620                                   
20621                                   set_volume_id:
20622 00001C91 52                                   push   dx              ; save registers
20623 00001C92 50                                   push   ax
20624 00001C93 E8F5FF                               call   haschange       ; does drive have changeline support?
20625 00001C96 740B                                 jz     short setvret   ; no, get out
20626 00001C98 E81000                               call   read_volume_id
20627 00001C9B 7209                                 jb     short seterr
20628 00001C9D E8A900                               call   transfer_volume_id ; copy the volume id     to special drive
20629 00001CA0 E8E2FF                               call   resetchanged    ; restore value       of change line
20630                                   setvret:
20631                                               ; 10/12/2022
20632                                               ; cf = 0
20633                                               ;clc                   ; no error, clear carry flag
20634 00001CA3 58                                   pop    ax              ; restore registers
20635 00001CA4 5A                                   pop    dx
20636 00001CA5 C3                                   retn
20637                                   ; ---------------------------------------------------------------------------
20638                                   
20639                                   seterr:
20640 00001CA6 5A                                   pop    dx              ; pop stack but don't overwrite ax
20641 00001CA7 5A                                   pop    dx              ; restore dx
20642 00001CA8 C3                                   retn
20643                                   ; ---------------------------------------------------------------------------
20644 00001CA9 0000                     root_sec:   dw 0                   ; root sector #
20645                                   
20646                                   ; 16/10/2022
20647                                   ;ROOTSEC equ root_sec - DOSBIOSEG_2C7h
20648                                   ; 09/12/2022
20649                                   ROOTSEC equ root_sec
20650                                   
20651                                   ; ============== S U B   R O U T I N E =======================================
20652                                   
20653                                   ; 16/10/2022
20654                                   
20655                                   ; read_volume_id read the volume id and tells if it has been changed.
20656                                   ;
20657                                   ;   on entry:
20658                                   ;   es:di points to current bds for drive.
20659                                   ;
20660                                   ;   on exit:
20661                                   ;   carry clear
20662                                   ;       si = 1  no change
20663                                   ;       si = 0  ?
20664                                   ;       si = -1 change
20665                                   ;
20666                                   ;   carry set:
20667                                   ;       error and ax has error code.
20668                                   
20669                                   read_volume_id:
20670 00001CAB 52                                   push   dx              ; preserve registers
20671 00001CAC 51                                   push   cx
20672 00001CAD 53                                   push   bx
20673 00001CAE 50                                   push   ax
20674 00001CAF 06                                   push   es              ; stack the bds last
20675 00001CB0 57                                   push   di
20676 00001CB1 1E                                   push   ds              ; point es to Bios_Data
20677 00001CB2 07                                   pop    es
20678 00001CB3 BF[4008]                             mov    di, tmp_vid     ; "NO NAME      "
20679 00001CB6 BE[6305]                             mov    si, nul_vid     ; "NO NAME      "
20680                                               ; 26/12/2023
20681 00001CB9 B90B00                               mov    cx, 11          ; PCDOS 7.1 - 02/09/2023
20682                                               ;mov   cx, 12          ; initialize tmp_vid to      null vi_id
20683                                   
20684                                               ;rep   movsb
20685                                               ; 26/12/2023
20686                                               ;rep movs byte ptr es:[di], byte ptr cs:[si]
20687                                               ;db 0FBh,2Eh,0A4h
20688                                               ;cs    ; nul_vid is in BIOSCODE segment
20689                                               ;rep movsb
20690 00001CBC F3                                   rep
20691 00001CBD 2E                                   cs
20692 00001CBE A4                                   movsb
20693                                   
20694 00001CBF 5F                                   pop    di
20695 00001CC0 07                                   pop    es
20696 00001CC1 268A450B                             mov    al, [es:di+11] ; [es:di+BDS.fats]
20697                                                                      ; # of fats
20698 00001CC5 268B4D11                             mov    cx, [es:di+17] ; [es:di+BDS.fatsecs]
20699                                                                      ; sectors / fat
20700 00001CC9 F6E1                                 mul    cl             ; size taken by       fats
```

```
00001CCB 26034509                       add     ax, [es:di+9]  ; [es:di+BDS.resectors]
                                                                ; add on reserved sectors
                                                                ;
                                                                ; ax is now sector # (0       based)
                                        ; 17/10/2022
00001CCF 2EA3[A91C]                     mov     [cs:ROOTSEC], ax
                                        ;mov    word ptr cs:198Fh, ax ; [cs:root_sec]
                                                                ; 0070h:3EFFh =       2C7h:198Fh
00001CD3 268B450C                       mov     ax, [es:di+12] ; [es:di+BDS.direntries]
                                                                ; # root dir entries
00001CD7 B104                           mov     cl, 4          ; 16 entries/sector
00001CD9 D3E8                           shr     ax, cl         ; divide by 16
                                        ;mov    cx, ax         ; cx is # of sectors to       scan
                                        ; 02/09/2023 (PCDOS 7.1, one byte opcode)
00001CDB 91                             xchg    ax, cx         ; cx is # of sectors to       scan
                                next_sec:
00001CDC 51                             push    cx             ; save outer loop counter
00001CDD 2EA1[A91C]                     mov     ax, [cs:ROOTSEC]
                                        ;mov    ax, word ptr cs:198Fh ; [cs:root_sec]
                                                                ; get sector #
00001CE1 268B4D13                       mov     cx, [es:di+19] ; [es:di+BDS.secpertrack]
                                                                ; sectors / track
00001CE5 31D2                           xor     dx, dx
00001CE7 F7F1                           div     cx

                                ; set up registers for call to read_sector

00001CE9 42                             inc     dx             ; dx= sectors into track
                                                                ; ax= track count from 0
00001CEA 88D1                           mov     cl, dl         ; sector to read
00001CEC 31D2                           xor     dx, dx
00001CEE 26F77515                       div     word [es:di+21] ; [es:di+BDS.heads]
                                                                ; # heads on this disc
00001CF2 88D6                           mov     dh, dl         ; head number
00001CF4 88C5                           mov     ch, al         ; track #
00001CF6 E8BDEB                         call    read_sector    ; get first sector of the root directory,
                                                                ; ds:bx -> directory sector
00001CF9 723F                           jb      short readviderr
00001CFB B91000                         mov     cx, 16         ; # of dir entries in a       block of root
00001CFE B008                           mov     al, 8          ; volume label bit
                                fvid_loop:
                                        ; 02/09/2023 (PCDOS 7.1)
00001D00 382F                           cmp     [bx], ch ; 0
                                        ;cmp    byte [bx], 0 ; end of dir?
00001D02 7433                           jz      short no_vid   ; yes, no vol id
00001D04 803FE5                         cmp     byte [bx], 0E5h ; empty entry?
00001D07 7405                           jz      short ent_loop ; yes, skip
00001D09 84470B                         test    [bx+11], al    ; is volume label bit set in fcb?
00001D0C 750F                           jnz     short found_vid       ; jmp yes
                                ent_loop:
00001D0E 83C320                         add     bx, 32         ; add length of       directory entry
00001D11 E2ED                           loop    fvid_loop
00001D13 59                             pop     cx             ; outer loop
00001D14 2EFF06[A91C]                   inc     word [cs:ROOTSEC]
                                        ;inc    word ptr cs:198Fh ; inc word [root_sec]
                                                                ; next sector
00001D19 E2C1                           loop    next_sec       ; continue
                                notfound:
                                        ; 02/09/2023
                                        ;xor    si, si
00001D1B EB13                           jmp     short fvid_ret
                                ; ---------------------------------------------------------------------------

                                found_vid:
                                        ; 02/09/2023
                                        ; cf = 0  ('test' instruction clears cf)
00001D1D 59                             pop     cx             ; clean stack of outer loop counter
00001D1E 89DE                           mov     si, bx         ; point to volume_id
00001D20 06                             push    es             ; preserve current bds
00001D21 57                             push    di
00001D22 1E                             push    ds
00001D23 07                             pop     es             ; point es to Bios_Data
00001D24 BF[4008]                       mov     di, tmp_vid    ; "NO NAME      "
00001D27 B90B00                         mov     cx, 11         ; VOLID_SIZ-1
                                                                ; length of string minus nul
00001D2A F3A4                           rep movsb              ; mov volume label to tmp_vid
                                        ;xor    al, al
                                        ; 02/09/2023
00001D2C 91                             xchg    ax, cx         ; ax = 0
00001D2D AA                             stosb                  ; null terminate
                                        ;;xor   si, si
                                        ; 02/09/2023
                                        ;xchg   ax, si         ; si = 0
00001D2E 5F                             pop     di             ; restore current bds
00001D2F 07                             pop     es
                                fvid_ret:
                                        ; 02/09/2023
00001D30 31F6                           xor     si, si ; 0

00001D32 58                             pop     ax
                                        ; 10/12/2022
                                        ; cf = 0
                                        ;clc
                                rvidret:
00001D33 5B                             pop     bx             ; restore registers
00001D34 59                             pop     cx
00001D35 5A                             pop     dx
00001D36 C3                             retn
                                ; ---------------------------------------------------------------------------

                                no_vid:
00001D37 59                             pop     cx             ; clean stack of outer loop counter
                                        ;jmp    short notfound ; not found
                                        ; 02/09/2023
00001D38 EBF6                           jmp     short fvid_ret
                                ; ---------------------------------------------------------------------------

                                readviderr:
00001D3A 5E                             pop     si             ; trash the outer loop counter
00001D3B 5E                             pop     si             ; caller's ax, return error code instead
00001D3C EBF5                           jmp     short rvidret

                                ; ---------------------------------------------------------------------------
                                        ; 26/12/2023 - Retro DOS v5.0
                                        ; 02/09/2023 - Retro DOS v4.2 (IO.SYS optimization)
                                        ; PCDOS 7.1 - IBMBIO.COM - BIOSCODE:1DCFh
                                preset_volid_addr:
00001D3E BE[4008]                       mov     si, tmp_vid    ; "NO NAME    "
                                        ; 26/12/2023
                                        ; PCDOS 7.1
00001D41 83C77D                         add     di, 125        ; BDS.volid
00001D44 B90B00                         mov     cx, 11         ; VOLID_SIZ (12 for MSDOS 5.0-6.22 versions)
                                        ; MSDOS 6.21 (MSDOS 5.0 & 6.?)
                                        ;add    di, 75         ; BDS.volid
```

```
20825                                           ;mov    cx, 12          ; VOLID_SIZ
20826                                           ;
20827 00001D47 FC                               cld
20828 00001D48 C3                               retn
20829
20830                                   ; =============== S U B   R O U T I N E =======================================
20831
20832                                   ; transfer_volume_id - copy the volume id from tmp to special drive
20833                                   ;
20834                                   ; inputs:   es:di has current bds
20835                                   ; outputs:  bds for drive has volume id from tmp
20836
20837                                           ; 27/12/2023 - Retro DOS v5.0
20838                                   transfer_volume_id:
20839 00001D49 57                               push    di              ; copy the volume id from tmp to special drive
20840                                           ;push   si
20841 00001D4A 51                               push    cx
20842                                           ; 27/12/2023
20843 00001D4B 56                               push    si
20844
20845                                           ;mov    si, tmp_vid     ; "NO NAME      "
20846                                           ;;add   di, BDS.volid
20847                                           ;add    di, 75          ; BDS.volid
20848                                           ;;mov   cx, VOLID_SIZ
20849                                           ;mov    cx, 12          ; VOLID_SIZ
20850                                           ;cld
20851                                           ; 02/09/2023 (PCDOS 7.1)
20852 00001D4C E8EFFF                           call    preset_volid_addr
20853
20854 00001D4F F3A4                             rep movsb
20855
20856                                           ; 27/12/2023
20857 00001D51 5E                               pop     si
20858                                   chk_volid_ok:
20859 00001D52 59                               pop     cx
20860                                           ;pop    si
20861 00001D53 5F                               pop     di
20862 00001D54 C3                               retn
20863
20864                                   ; =============== S U B   R O U T I N E =======================================
20865
20866                                   ;  check_volume_id - compare volume id in tmp area with
20867                                   ;               one expected for drive
20868                                   ;
20869                                   ;   inputs: es:di has current bds for drive
20870                                   ;   outputs:       zero true means it matched
20871
20872                                           ; 27/12/2023 - Retro DOS v5.0
20873                                   check_volume_id:
20874 00001D55 57                               push    di
20875 00001D56 51                               push    cx
20876
20877                                           ;mov    si, tmp_vid     ; "NO NAME      "
20878                                           ;;add   di, BDS.volid
20879                                           ;add    di, 75          ; BDS.volid
20880                                           ;;mov   cx, VOLID_SIZ
20881                                           ;mov    cx, 12          ; VOLID_SIZ
20882                                           ;cld
20883                                           ; 02/09/2023 (PCDOS 7.1)
20884 00001D57 E8E4FF                           call    preset_volid_addr
20885
20886 00001D5A F3A6                             repe cmpsb              ; are the 2 volume_ids the same?
20887
20888                                           ; 27/12/2023
20889                                           ;pop    cx
20890                                           ;pop    di
20891                                           ;retn
20892 00001D5C EBF4                             jmp     short chk_volid_ok
20893
20894                                   ; =============== S U B   R O U T I N E =======================================
20895
20896                                   ;   fat_check - see of the fatid has changed in the specified drive.
20897                                   ;         - uses the fat id obtained from the boot sector.
20898                                   ;
20899                                   ;   inputs: medbyt is expected fat id
20900                                   ;           es:di points to current bds
20901                                   ;
20902                                   ;   output: si = -1 if fat id different,
20903                                   ;           si = 0 otherwise
20904                                   ;
20905                                   ;   no other registers changed.
20906
20907                                   fat_check:
20908 00001D5E 50                               push    ax
20909 00001D5F 31F6                             xor     si, si          ; say fat id's are same.
20910 00001D61 A0[1F01]                         mov     al, [medbyt]    ; 19/10/2022
20911 00001D64 263A4510                         cmp     al, [es:di+10h]         ; [es:di+BDS.media]
20912                                                                   ; compare it with the bds medbyte
20913 00001D68 7401                             jz      short okret1    ; carry clear
20914 00001D6A 4E                               dec     si
20915                                   okret1:
20916 00001D6B 58                               pop     ax
20917 00001D6C C3                               retn
20918
20919                                   ; ---------------------------------------------------------------------------
20920
20921                                   ; BIOSCODE:1DFEh (PCDOS 7.1 IBMBIO.COM) ; 27/12/2023
20922                                           ;times 2 db 0
20923
20924                                   ; BIOSCODE:1A69h (MSDOS 6.21 IO.SYS) ((& MSDOS 6.22 IO.SYS))
20925                                           ;times 7 db 0
20926
20927                                   ; BIOSCODE:180Bh (MSDOS 5.0 IO.SYS)
20928
20929                                           ; 09/12/2022
20930                                           ;times 4 db 0  ; 17/10/2022
20931                                           ;db 4 dup(0)   ; times 4 db 0
20932
20933                                   ; ---------------------------------------------------------------------------
20934
20935                                           ; 09/12/2022
20936                                           ;db 0
20937
20938                                   number2div  equ ($-BCode_start)
20939                                   number2mod  equ (number2div % 16)
20940
20941                                   %if number2mod>0 & number2mod<16
20942 00001D6D 00<rep 3h>                       times (16-number2mod) db 0
20943                                   %endif
20944
20945                                   ;align 16
20946
20947                                   ; 09/12/2022
20948                                   BCODE_END   equ $ - BCode_start
```

```
                        ; 31/03/2024
BCODEEND:
;SYSINITSEG equ IOSYSCODESEG+(BCODE_END>>4)
; 13/12/2022
SYSINITOFFSET      equ BCODE_END
SYSINITSEG  equ IOSYSCODESEG+(SYSINITOFFSET>>4)

; 30/12/2022 - Retro DOS v4.2
; (SYSINITSEG is 473h for MSDOS 6.21 IO.SYS)

;--- End of DOSBIOS code segment ---------------------------------------------

; 16/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
; 01/05/2019 - Retro DOS v4.0
; ============================================================================
; end of BIOSCODE

; ----------------------------------------------------------------------------
; %include sysinit5.s     ; 09/12/2022
; ----------------------------------------------------------------------------

;============================================================================
; (IO.SYS) SYSINIT SEGMENT
;============================================================================
; 09/12/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
; 30/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
;
; 27/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)

section .SYSINIT vstart=0

; ****************************************************************************
; SYSINIT.BIN (MSDOS 6.21 IO.SYS) - RETRO DOS v4.0 by ERDOGAN TAN - 21/10/2022
; ----------------------------------------------------------------------------
; Last Update: 04/01/2023 (Modified IO.SYS)  ((Previous: 30/12/2022))
; ----------------------------------------------------------------------------
; Beginning: 03/06/2018 (Retro DOS 3.0), 21/03/2019 (Retro DOS 4.0)
; ----------------------------------------------------------------------------
; Assembler: NASM version 2.15
; ----------------------------------------------------------------------------
;      ((nasm sysinit6.s -l sysinit6.lst -o SYSINIT6.BIN -Z error.txt))
; ----------------------------------------------------------------------------
; Modified from 'sysinit2.s' (SYSINIT2.BIN) file of Retro DOS v3.0 (6/7/2018)
; ----------------------------------------------------------------------------
; Derived from 'SYSINIT1.ASM' and 'SYSINIT2.ASM' files of MSDOS 6.0
; source code by Microsoft, 1991
; ----------------------------------------------------------------------------
; Derived from 'SYSINIT.ASM' file of MSDOS 2.0 (IBM PCDOS v2.0) source code
; by Microsoft, 12/10/1983
; ****************************************************************************
; main file: 'retrodos4.s'
; incbin 'SYSINIT3.BIN' ; (SYINITSEG)

; 30/12/2022 - Retro DOS v4.2
; Retro DOS v4.0 - 2019
; SYSINIT (MSDOS 6.21 IO.SYS) draft: 'sysinit3.s' (01/07/2019)

; 21/10/2022
; ----------------------------------------------------------------------------
; This source code (version) is based on SYSINIT source code of disassembled
; MSDOS 5.0 IO.SYS file (SYSINIT.BIN)
; Dissassembler: Hex-Rays Interactive Disassembler (IDA)
; ----------------------------------------------------------------------------
; Binary file splitter & joiner: FFSJ v3.3

;----------------------------------------------------------------
; SYSINIT.TXT (27/01/1983)
;----------------------------------------------------------------
;    SYSINIT is  a module linked behind the OEM bios.  It takes
;over  the  system  initialization  after  the  OEM  bios   has
;performed any  initialization  it  needs  to  do.   Control is
;transfered with a long jump to the external  variable  SYSINIT
;
;
;   The OEM  has  the  following  variables declared external:
;
;   CURRENT_DOS_LOCATION    WORD
;
;This word  contains  the  segment  number of the DOS before it
;is relocated.  The OEM bios must set this value.
;
;   FINAL_DOS_LOCATION      WORD
;
;This word contains the segment number of the DOS after SYSINIT
;moves it.  The OEM bios must set this value.
;
;   DEVICE_LIST             DWORD
;
;This  double  word  pointer  points  to  the  linked  list  of
;character and block device drivers.  The  OEM  must  set  this
;value.
;
;   MEMORY_SIZE             WORD
;
;This word  contains  the  number  of  RAM  paragraphs.  If the
;bios doesn't set  this  variable  SYSINIT  will  automatically
;calculate it.   NOTE:  systems with PARITY checked memory must
;size memory in the BIOS.  SYSINITs method is to  write  memory
;and read it back until it gets a mismatch.
;
;   DEFAULT_DRIVE           BYTE
;
;This is  the initial default drive when the system first comes
;up.  drive a=0, drive b=1,  etc.   If  the  bios  doesn't  set
;it then drive a is assumed.
;
;   BUFFERS                 BYTE
;
;This is  the  default  number of buffers for the system.  This
;value may be overridden by the user in  the  CONFIG.SYS  file.
;It is DBed to 2 in SYSINIT it should be greater than 1.
;
;   FILES                   BYTE
;
;This is  the  default  number  of  files for the system.  This
;value may be overridden by the user in  the  CONFIG.SYS  file.
;It is  DBed  to  8 in SYSINIT, values less than 5 are ignored.
;
;   SYSINIT                 FAR
;
;The entry  point  of  the  SYSINIT  module.  OEM BIOS jumps to
;this label at the end of its INIT code.
;
;   The OEM  has  the  following  variables declared public:
```

```
                        ;
                        ;   RE_INIT                 FAR
                        ;
                        ;This is an entry point which allows the BIOS to do some INIT
                        ;work  after  the  DOS is initialized.  ALL REGISTERS MUST BE
                        ;PRESERVED.  On entry DS points to the first available memory
                        ;(after  the DOS).  DS:0 points to a 100H byte program header
                        ;prefix which represents  the  "program"  currently  running.
                        ;This  program  should  be  thought  of  as  the OEM BIOS and
                        ;SYSINIT taken together.  This is not  a  normal  program  in
                        ;that  no  memory  is  allocated to it, it is running in free
                        ;memory.
                        ;NOTES:
                        ;     At the time this routine is called SYSINIT occupies the
                        ;highest 10K of memory ("highest" is determined by the  value
                        ;of the MEMORY_SIZE variable), DO NOT DO WRITES THERE.
                        ;     Since this is called AFTER DOS is initialized, you can
                        ;make system calls.  This also implies that the code for this
                        ;routine    CANNOT   be   thrown   away   by   use   of   the
                        ;FINAL_DOS_LOCATION since the DOS has already been moved.
                        ;     If you don't want  anything done just set this to point
                        ;at a FAR RET instruction.

                        ; --------------------------------------------------------------------------
                        ; TITLE   BIOS SYSTEM INITIALIZATION
                        ; --------------------------------------------------------------------------

                        ;include version.inc
                        ; --------------------------------------------------------------------------

                        ;FALSE   EQU     0
                        ;TRUE    EQU     0FFFFh

                        ;IBMVER         EQU     TRUE
                        ;IBMCOPYRIGHT EQU   FALSE
                        ;STACKSW    EQU        TRUE             ;Include Switchable Hardware Stacks
                        ;IBMJAPVER  EQU     FALSE         ; If TRUE set KANJI true also
                        ;MSVER      EQU     FALSE
                        ;ALTVECT    EQU     FALSE         ; Switch to build ALTVECT version
                        ;KANJI      EQU     FALSE

                        ;(MSDOS 6.0, versiona.inc, 1991)
                        ; --------------------------------------------------------------------------
                        ;MAJOR_VERSION  EQU       6
                        ;;MINOR_VERSION    EQU     0       ;6.00
                        ;MINOR_VERSION  EQU       21      ;6.21  ; 21/03/2019 - Retro DOS v4.0

                        ; 22/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0)
                        ; --------------------------------------------------------------------------
                        ;MAJOR_VERSION   EQU      5
                        ;MINOR_VERSION   EQU      0

                        ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21)
                        ;MAJOR_VERSION   EQU      6
                        ;MINOR_VERSION   EQU      22

                        ; 21/02/2024 - Retro DOS v5.0 (Modified PCDOS 7.1)
                        MAJOR_VERSION   EQU       7
                        MINOR_VERSION   EQU       10

                        expected_version equ      (MINOR_VERSION<<8)+MAJOR_VERSION

                        ;DOSREVNM equ      00000000b       ; m037 - bits 0-2 = revision number of DOS
                                                           ; currently 0.
                        DOSREVNM equ       00000111b      ; [[[ 7 for Retro DOS v4.0 ]]] (21/03/2019)
                        DOSINROM equ       00001000B       ; bit 3 of ver flags returned in BH
                        DOSINHMA equ       00010000B       ; bit 4 of ver flags

                        ;       if1
                        ;       %OUT  ... for DOS Version 5.00 ...
                        ;       endif

                               ;*******************************
                               ;Each assembler program should:
                               ;  mov ah,030h                  ;DOS Get Version function
                               ;  int 021h                     ;Version ret. in AX,minor version first
                               ;  cmp ax,expected_version      ;ALL utilities should check for an
                               ;  jne error_handler            ; EXACT version match.
                               ;*******************************

                        ; --------------------------------------------------------------------------
                        ; device definitions

                        ;Attribute bit masks
                        DEVTYP  EQU     8000h           ;Bit 15 - 1  if Char, 0 if block
                        DEVIOCTL EQU    4000h           ;Bit 14 - CONTROL mode bit
                        ISFATBYDEV EQU  2000h           ;Bit 13 - Device uses FAT ID bytes, comp media.
                        ISCIN   EQU     0001h           ;Bit 0 - This device is the console input.
                        ISCOUT  EQU     0002h           ;Bit 1 - This device is the console output.
                        ISNULL  EQU     0004h           ;Bit 2 - This device is the null device.
                        ISCLOCK EQU     0008h           ;Bit 3 - This device is the clock device.
                        ISIBM   EQU     0010h           ;Bit 4 - This device is special

                        ; The device table list has the form:
                        struc       SYSDEV
00000000 ????????       .NEXT:              resd 1          ;Pointer to next device header
00000004 ????           .ATT:               resw 1          ;Attributes of the device
00000006 ????           .STRAT:             resw 1          ;Strategy entry point
00000008 ????           .INT:               resw 1          ;Interrupt entry point
0000000A ????????????????      .NAME:              resb 8          ;Name of device (only first byte used for block)
                        .size:
                        endstruc

                        ;Static Reguest Header
                        struc       SRHEAD
00000000 ??             .REQLEN:    resb 1          ;Length in bytes of request block
00000001 ??             .REQUNIT:   resb 1          ;Device unit number
00000002 ??             .REQFUNC:   resb 1          ;Type of request
00000003 ????           .REQSTAT:   resw 1          ;Status Word
00000005 ????????????????              resb 8          ;Reserved for queue links
                        .size:
                        endstruc

                        ;Status word masks
                        STERR   EQU     8000H           ;Bit 15 - Error
                        STBUI   EQU     0200H           ;Bit 9 - Buisy
                        STDON   EQU     0100H           ;Bit 8 - Done
                        STECODE EQU     00FFH           ;Error code
                        WRECODE EQU     0

                        ;Function codes
                        DEVINIT EQU     0               ;Initialization
                        DINITHL EQU     26              ;Size of init header
                        DEVMDCH EQU     1               ;Media check
```

```
21197                                   DMEDHL  EQU     15              ;Size of media check header
21198                                   DEVBPB  EQU     2               ;Get BPB
21199                                   DEVRDIOCTL EQU  3               ;IOCTL read
21200                                   DBPBHL  EQU     22              ;Size of Get BPB header
21201                                   DEVRD   EQU     4               ;Read
21202                                   DRDWRHL EQU     22              ;Size of RD/WR header
21203                                   DEVRDND EQU     5               ;Non destructive read no wait (character devs)
21204                                   DRDNDHL EQU     14              ;Size of non destructive read header
21205                                   DEVIST  EQU     6               ;Input status
21206                                   DSTATHL EQU     13              ;Size of status header
21207                                   DEVIFL  EQU     7               ;Input flush
21208                                   ; 21/02/2024
21209                                   ;DFLSHL EQU     15              ;Size of flush header
21210                                   DFLSHL      equ    13      ; PCDOS 7.1 IBMDOS.COM  ; 21/02/2024
21211                                   DEVWRT  EQU     8               ;Write
21212                                   DEVWRTV EQU     9               ;Write with verify
21213                                   DEVOST  EQU     10              ;Output status
21214                                   DEVOFL  EQU     11              ;Output flush
21215                                   DEVWRIOCTL EQU  12              ;IOCTL write
21216
21217                                   ; ----------------------------------------------------------------------
21218                                   struc       SYS_FCB
21219 00000000 ??                      .fcb_drive: resb 1
21220 00000001 ????????????????        .fcb_name:  resb 8
21221 00000009 ??????                  .fcb_ext:   resb 3
21222 0000000C ????                    .fcb_EXTENT:       resw 1
21223 0000000E ????                    .fcb_RECSIZ:       resw 1  ; Size of record (user settable)
21224 00000010 ????                    .fcb_FILSIZ:       resw 1  ; Size of file in bytes; used with the following
21225                                                              ; word
21226 00000012 ????                    .fcb_DRVBP: resw 1  ; BP for SEARCH FIRST and SEARCH NEXT
21227 00000014 ????                    .fcb_FDATE: resw 1  ; Date of last writing
21228 00000016 ????                    .fcb_FTIME: resw 1  ; Time of last writing
21229 00000018 ??                      .fcb_DEVID: resb 1  ; Device ID number, bits 0-5 if file.
21230                                                              ; bit 7=0 for file, bit 7=1 for I/O device
21231                                                              ; If file, bit 6=0 if dirty
21232                                                              ; If I/O device, bit 6=0 if EOF (input)
21233                                                              ;               Bit 5=1 if Raw mode
21234                                                              ;               Bit 0=1 if console input device
21235                                                              ;               Bit 1=1 if console output device
21236                                                              ;               Bit 2=1 if null device
21237                                                              ;               Bit 3=1 if clock device
21238 00000019 ????                  .fcb_FIRCLUS:      resw 1  ; First cluster of file
21239 0000001B ????                  .fcb_CLUSPOS:      resw 1  ; Position of last cluster accessed
21240 0000001D ????                  .fcb_LSTCLUS:      resw 1  ; Last cluster accessed and directory
21241 0000001F ??                                       resb 1  ; pack 2 12 bit numbers into 24 bits...
21242 00000020 ??                    .fcb_NR:    resb 1  ; Next record
21243 00000021 ????????              .fcb_RR:    resb 4  ; Random record
21244                                 .size:
21245                                 endstruc
21246
21247                                 ; ----------------------------------------------------------------------
21248                                 ; Field definition for I/O buffer information
21249
21250                                 ; 22/03/2019 - Retro DOS v4.0 (MSDOS 6.0, BUFFER.INC, 1991)
21251
21252                                     ; 03/01/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMDOS.COM)
21253
21254                                 struc BUFFINFO
21255 00000000 ????                  .buf_next:  resw 1          ; Pointer to next buffer in list
21256 00000002 ????                  .buf_prev:  resw 1          ; Pointer to prev buffer in list
21257 00000004 ??                    .buf_ID:    resb 1          ; Drive of buffer (bit 7 = 0)
21258                                                            ; SFT table index (bit 7 = 1)
21259                                                            ; = FFH if buffer free
21260 00000005 ??                    .buf_flags: resb 1          ; Bit 7 = 1 if Remote file buffer
21261                                                            ;       = 0 if Local device buffer
21262                                                            ; Bit 6 = 1 if buffer dirty
21263                                                            ; Bit 5 = Reserved
21264                                                            ; Bit 4 = Search bit (bit 7 = 1)
21265                                                            ; Bit 3 = 1 if buffer is DATA
21266                                                            ; Bit 2 = 1 if buffer is DIR
21267                                                            ; Bit 1 = 1 if buffer is FAT
21268                                                            ; Bit 0 = Reserved
21269 00000006 ????????              .buf_sector:       resd 1          ; Sector number of buffer (flags bit 7 = 0)
21270                                 ; The next two items are often refed as a word (flags bit 7 = 0)
21271 0000000A ??                    .buf_wrtcnt:       resb 1          ; For FAT sectors, # times sector written out
21272 0000000B ????                  .buf_wrtcntinc:    resw 1          ; "   "      "   , # sectors between each write
21273 0000000D ????                              resw 1 ; * ; 03/01/2024 ; PCDOS 7.1
21274                                                       ; hw of sectors per FAT
21275 0000000F ????????              .buf_DPB:   resd 1          ; Pointer to drive parameters
21276 00000013 ????                  .buf_fill:  resw 1          ; How full buffer is (flags bit 7 = 1)
21277 00000015 ??                    .buf_reserved:     resb 1          ; make DWORD boundary for 386
21278 00000016 ????                              resw 1 ; * ; 03/01/2024 ; PCDOS 7.1
21279                                                       ; reserved word for dword boundary
21280                                 .size:      ; 20 bytes ; MSDOS 5.0 to 6.22
21281                                     ; 24 bytes ; PCDOS 7.1 ; 03/01/2024
21282                                 endstruc
21283
21284                                 %define buf_offset BUFFINFO.buf_sector ; 22/07/2019
21285                                                    ;For buf_flags bit 7 = 1, this is the byte
21286                                                    ;offset of the start of the buffer in
21287                                                    ;the file pointed to by buf_ID. Thus
21288                                                    ;the buffer starts at location
21289                                                    ;buf_offset in the file and contains
21290                                                    ;buf_fill bytes.
21291
21292                                 bufinsiz    equ     BUFFINFO.size  ; Size of structure in bytes
21293
21294
21295                                 buf_Free    equ     0FFh           ; buf_id of free buffer
21296
21297                                 ;Flag byte masks
21298                                 buf_isnet   EQU     10000000B
21299                                 buf_dirty   EQU     01000000B
21300                                 ;***
21301                                 buf_visit   EQU     00100000B
21302                                 ;***
21303                                 buf_snbuf   EQU     00010000B
21304
21305                                 buf_isDATA  EQU     00001000B
21306                                 buf_isDIR   EQU     00000100B
21307                                 buf_isFAT   EQU     00000010B
21308                                 buf_type_0  EQU     11110001B      ; AND sets type to "none"
21309
21310                                 buf_NetID   EQU     bufinsiz
21311
21312                                 ; ----------------------------------------------------------------------
21313
21314                                 ; ----------------------------------------------------------------------
21315                                 ;** DPB - Drive Parameter Block
21316
21317                                 ; 25/03/2019 - Retro DOS v4.0 (MSDOS 6.0, DPB.INC, 1991)
21318
21319                                 ;   BUGBUG - this isn't authorative - it's my probably incomplete and
21320                                 ;   possibly inaccurate deductions from code study... - jgl
```

```
                                ;
                                ;   The DPB is DOS's main structure for describing block devices.
                                ;   It contains info about the "Drive" intermingled with info about
                                ;   the FAT file system which is presumably on the drive.  I don't know
                                ;   how those fields are used if it's not the FAT file system - BUGBUG
                                ;
                                ;   The DPBs are statically allocated and chained off of DPBHead.
                                ;   Users scan this chain looking for a match on DPB_DRIVE.
                                ;   The DPBs are built at init time from info in the SYSDEV structure.

                               ; 08/06/2018 - Retro DOS v3.0 (MSDOS 3.3, DPB.INC, 24/07/1987)

                               ; 12/05/2019 - Retro DOS v4.0

                               ; 01/01/2024
                               %if 0

                               struc      DPB
                               .DRIVE:            resb 1         ; Logical drive # assoc with DPB (A=0,B=1,...)
                               .UNIT:             resb 1         ; Driver unit number of DPB
                               .SECTOR_SIZE:      resw 1         ; Size of physical sector in bytes
                               .CLUSTER_MASK:     resb 1         ; Sectors/cluster - 1
                               .CLUSTER_SHIFT:    resb 1         ; Log2 of sectors/cluster
                               .FIRST_FAT: resw 1        ; Starting record of FATs
                               .FAT_COUNT: resb 1        ; Number of FATs for this drive
                               .ROOT_ENTRIES:     resw 1         ; Number of directory entries
                               .FIRST_SECTOR:     resw 1         ; First sector of first cluster
                               .MAX_CLUSTER:      resw 1         ; Number of clusters on drive + 1
                               ; MSDOS 3.3
                               ;.FAT_SIZE: resb 1         ; Number of records occupied by FAT
                               ; MSDOS 6.0
                               .FAT_SIZE:  resw 1         ; Number of records occupied by FAT
                               .DIR_SECTOR:       resw 1         ; Starting record of directory
                               .DRIVER_ADDR:      resd 1         ; Pointer to driver
                               .MEDIA:            resb 1         ; Media byte
                               .FIRST_ACCESS:     resb 1         ; This is initialized to -1 to force a media
                                                          ; check the first time this DPB is used
                               .NEXT_DPB:  resd 1         ; Pointer to next Drive parameter block
                               .NEXT_FREE: resw 1         ; Cluster # of last allocated cluster
                               .FREE_CNT:  resw 1         ; Count of free clusters, -1 if unknown
                               .size:
                               endstruc

                               %else

                               ; 01/01/2024 - Retro DOS v5.0 (PCDOS 7.1)

                               struc      DPB
00000000 ??                    .DRIVE:            resb 1  ; 0     ; Logical drive # assoc with DPB (A=0,B=1,...)
00000001 ??                    .UNIT:             resb 1  ; 1     ; Driver unit number of DPB
00000002 ????                  .SECTOR_SIZE:      resw 1  ; 2     ; Size of physical sector in bytes
00000004 ??                    .CLUSTER_MASK:     resb 1  ; 4     ; Sectors/cluster - 1
00000005 ??                    .CLUSTER_SHIFT:    resb 1  ; 5     ; Log2 of sectors/cluster
00000006 ????                  .FIRST_FAT: resw 1  ; 6     ; Starting record of FATs
00000008 ??                    .FAT_COUNT: resb 1  ; 8     ; Number of FATs for this drive
00000009 ????                  .ROOT_ENTRIES:     resw 1  ; 9     ; Number of directory entries
0000000B ????                  .FIRST_SECTOR:     resw 1  ; 11    ; First sector of first cluster
0000000D ????                  .MAX_CLUSTER:      resw 1  ; 13    ; Number of clusters on drive + 1
0000000F ????                  .FAT_SIZE:  resw 1  ; 15    ; Number of records occupied by FAT
00000011 ????                  .DIR_SECTOR:       resw 1  ; 17    ; Starting record of directory
00000013 ????????              .DRIVER_ADDR:      resd 1  ; 19    ; Pointer to driver
00000017 ??                    .MEDIA:            resb 1  ; 23    ; Media byte
00000018 ??                    .FIRST_ACCESS:     resb 1  ; 24    ; This is initialized to -1 to force a media
                                                          ;       ; check the first time this DPB is used
00000019 ????????              .NEXT_DPB:  resd 1  ; 25    ; Pointer to next Drive parameter block
0000001D ????                  .NEXT_FREE: resw 1  ; 29    ; Cluster # of last allocated cluster
0000001F ????                  .FREE_CNT:  resw 1  ; 31    ; Count of free clusters, -1 if unknown
                               ; FAT32 fs ; 01/01/2024
                               ; ref: https://en.wikibooks.org/wiki/
                               ;      First_steps_towards_system_programming_under_MS-DOS_7/Appendix
                               ;   -- A.03-1. Structure of Drive Parameters Blocks (DPB) ---
00000021 ????                  .FREE_CNT_HW:      resw 1  ; 33    ; High word of free cluster count
00000023 ????                  .EXT_FLAGS: resw 1  ; 35    ; FAT32 extended flags (active FAT number)
00000025 ????                  .FSINFO_SECTOR:    resw 1  ; 37    ; (FAT32 fs) FSINFO structure sector address
00000027 ????                  .BKBOOT_SECTOR:    resw 1  ; 39    ; (FAT32 fs) Backup Boot Sector address
00000029 ????????              .FCLUS_FSECTOR: resd 1     ; 41    ; The first cluster's first sector address
0000002D ????????              .LAST_CLUSTER:     resd 1  ; 45    ; The last cluster number
00000031 ????????              .FAT32_SIZE:       resd 1  ; 49    ; Number of FAT sectors (for FAT32 fs)
00000035 ????????              .ROOT_CLUSTER:     resd 1  ; 53    ; Root directory's cluster number (FAT32 fs)
                               ; 01/01/2024 - Retro DOS v5.0
00000039 ????????              .FAT32_NXTFREE:    resd 1  ; 57    ; The next free cluster (for FAT32 fs)
                               .size:             ; 61 bytes ; 01/01/2024 (PCDOS 7.1)
                               endstruc

                               %endif

                               DPBSIZ  EQU     DPB.size   ; Size of the structure in bytes

                               DSKSIZ  EQU DPB.MAX_CLUSTER       ; Size of disk (temp used during init only)

                               ; ---------------------------------------------------------------------------
                               ; 26/03/2018

                               ; IOCTL SUB-FUNCTIONS
                               IOCTL_GET_DEVICE_INFO      EQU     0
                               IOCTL_SET_DEVICE_INFO      EQU     1
                               IOCTL_READ_HANDLE  EQU     2
                               IOCTL_WRITE_HANDLE EQU     3
                               IOCTL_READ_DRIVE   EQU     4
                               IOCTL_WRITE_DRIVE  EQU     5
                               IOCTL_GET_INPUT_STATUS     EQU     6
                               IOCTL_GET_OUTPUT_STATUS EQU       7
                               IOCTL_CHANGEABLE?  EQU     8
                               IOCTL_SHARING_RETRY        EQU     11
                               GENERIC_IOCTL_HANDLE       EQU     12
                               GENERIC_IOCTL              EQU     13

                               ; GENERIC IOCTL SUB-FUNCTIONS
                               RAWIO                      EQU     8

                               ; RAWIO SUB-FUNCTIONS
                               GET_DEVICE_PARAMETERS      EQU     60H
                               SET_DEVICE_PARAMETERS      EQU     40H
                               READ_TRACK         EQU     61H
                               WRITE_TRACK        EQU     41H
                               VERIFY_TRACK               EQU     62H
                               FORMAT_TRACK               EQU     42H

                               ; DEVICETYPE VALUES
                               MAX_SECTORS_IN_TRACK       EQU     63
                               DEV_5INCH          EQU     0
                               DEV_5INCH96TPI             EQU     1
                               DEV_3INCH720KB             EQU     2
                               DEV_8INCHSS        EQU     3
```

```
                                DEV_8INCHDS         EQU     4
                                DEV_HARDDISK                EQU     5
                                DEV_OTHER           EQU     7
                                ;DEV_3INCH1440KB    EQU     7
                                DEV_3INCH2880KB             EQU     9
                                ; Retro DOS v2.0 - 26/03/2018
                                ;;DEV_TAPE          EQU     6
                                ;;DEV_ERIMO         EQU     8
                                ;DEV_3INCH2880KB    EQU     9
                                DEV_3INCH1440KB             EQU     10

                                ;MAX_DEV_TYPE               EQU     9       ; MAXIMUM DEVICE TYPE THAT WE
                                                                            ; CURRENTLY SUPPORT.
                                MAX_DEV_TYPE                EQU     10

                                struc A_SECTORTABLE
00000000 ????                   .ST_SECTORNUMBER:  resw    1
00000002 ????                   .ST_SECTORSIZE:            resw    1
                                .size:
                                endstruc

                                ; 25/03/2019 - Retro DOS v4.0  (MSDOS 6.0, BPB.INC, IOCTL.INC)

                                ;** BIOS PARAMETER BLOCK DEFINITION
                                ;
                                ;   The BPB contains information about the disk structure.  It dates
                                ;   back to the earliest FAT systems and so FAT information is
                                ;   intermingled with physical driver information.
                                ;
                                ;   A boot sector contains a BPB for its device; for other disks
                                ;   the driver creates a BPB.  DOS keeps copies of some of this
                                ;   information in the DPB.
                                ;
                                ;   The BDS structure contains a BPB within it.

                                ; 01/01/2024
                                %if 0

                                struc A_BPB
                                .BPB_BYTESPERSECTOR:       resw    1
                                .BPB_SECTORSPERCLUSTER:    resb    1
                                .BPB_RESERVEDSECTORS:      resw    1
                                .BPB_NUMBEROFFATS: resb    1
                                .BPB_ROOTENTRIES:  resw    1
                                .BPB_TOTALSECTORS: resw    1
                                .BPB_MEDIADESCRIPTOR:      resb    1
                                .BPB_SECTORSPERFAT:        resw    1
                                .BPB_SECTORSPERTRACK:      resw    1
                                .BPB_HEADS:         resw    1
                                .BPB_HIDDENSECTORS:        resw    1
                                                    resw    1
                                .BPB_BIGTOTALSECTORS:      resw    1
                                                    resw    1
                                                    resb    6       ; NOTE:  many times these
                                ;                                   ;       6 bytes are omitted
                                ;                                   ;       when BPB manipulations
                                ;                                   ;       are performed!
                                .size:
                                endstruc

                                %else

                                ; 14/04/2024
                                ; 01/01/2024 - Retro DOS v5.0

                                struc A_BPB
00000000 ????                   .BYTESPERSECTOR:    resw 1
00000002 ??                     .SECTORSPERCLUSTER: resb 1
00000003 ????                   .RESERVEDSECTORS:   resw 1
00000005 ??                     .NUMBEROFFATS:         resb 1
00000006 ????                   .ROOTENTRIES:          resw 1
00000008 ????                   .TOTALSECTORS:         resw 1
0000000A ??                     .MEDIADESCRIPTOR:   resb 1
0000000B ????                   .SECTORSPERFAT:        resw 1
0000000D ????                   .SECTORSPERTRACK:   resw 1
0000000F ????                   .HEADS:                resw 1
00000011 ????????               .HIDDENSECTORS:        resd 1
00000015 ????????               .BIGTOTALSECTORS:   resd 1
                                ;............ FAT32 ......  + 28
00000019 ????????               .FATSIZE32:     resd 1
0000001D ????                   .EXTFLAGS:      resw 1
0000001F ????                   .FSVER:                resw 1
00000021 ????????               .ROOTDIRCLUSTER:    resd 1
00000025 ????                   .FSINFOSECTOR:         resw 1  ; (offset from FAT32 bs)
00000027 ????                   .BACKUPBOOTSECTOR:  resw 1  ; (offset from FAT32 bs)
00000029 <res Ch>               .RESERVEDBYTES:        resb 12 ; (zero bytes)
                                                ; 14/04/2024
00000035 ???????????                           resb 6  ; A_BPB.size must be 59
                                .size:
                                endstruc

                                %endif

                                struc A_DEVICEPARAMETERS
00000000 ??                     .DP_SPECIALFUNCTIONS:      resb    1
00000001 ??                     .DP_DEVICETYPE:            resb    1
00000002 ????                   .DP_DEVICEATTRIBUTES:      resw    1
00000004 ????                   .DP_CYLINDERS:             resw    1
00000006 ??                     .DP_MEDIATYPE:             resb    1
00000007 <res 3Bh>              .DP_BPB:           resb    A_BPB.size
00000042 ????                   .DP_TRACKTABLEENTRIES:     resw    1
00000044 <res FCh>              .DP_SECTORTABLE:   resb    MAX_SECTORS_IN_TRACK * A_SECTORTABLE.size
                                endstruc

                                ; ----------------------------------------------------------------------------
                                ; structure, equates for devmark for mem command.

                                ; 25/03/2019 - Retro DOS v4.0 (MSDOS 6.0, DEVMARK.INC, 1991)

                                struc devmark
00000000 ??                      .id:           resb 1
00000001 ????                    .seg:          resw 1
00000003 ????                    .size:         resw 1
00000005 ??????                  .dum:          resb 3
00000008 ????????????????       .filename: resb 8
                                endstruc

                                devmark_stk equ     'S'
                                devmark_device      equ     'D'
                                devmark_ifs equ     'I'
                                devmark_buf equ     'B'
                                devmark_cds equ     'L' ; lastdrive
                                devmark_files       equ     'F'
                                devmark_fcbs        equ     'X'
```

```
                                devmark_inst       equ     'T' ; used for sysinit base for install= command.
                                devmark_ems_stub equ       'E'

                                setbrkdone  equ     00000001b
                                for_devmark equ     00000010b
                                not_for_devmark equ       11111101b

                                ; --------------------------------------------------------------------------
                                ; Memory arena structure

                                ; 24/03/2019 - Retro DOS v4.0
                                ; (MSDOS 6.0, ARENA.INC)

                                ;** Arena Header

                                struc ARENA
00000000 ??                     .SIGNATURE: resb 1         ; 4D for valid item, 5A for last item
00000001 ????                   .OWNER:             resw 1         ; owner of arena item
00000003 ????                   .SIZE:              resw 1         ; size in paragraphs of item
00000005 ??????                 .RESERVED   resb 3         ; reserved
00000008 ????????????????       .NAME:              resb 8         ; owner file name
                                endstruc

                                ; 12/04/2019

                                arena_owner_system EQU 0   ; free block indication

                                arena_signature_normal     EQU 4Dh ; valid signature, not end of arena
                                arena_signature_end        EQU 5Ah ; valid signature, last block in arena

                                ; --------------------------------------------------------------------------
                                ; Process data block (otherwise known as program header)

                                ; 23/03/2019 - Retro DOS v4.0

                                ; (MSDOS 6.0 - PDB.INC, 1991)

                                FILPERPROC  EQU     20

                                struc PDB   ; Process_data_block
00000000 ????                   .EXIT_CALL: resw 1         ; INT int_abort system terminate
00000002 ????                   .BLOCK_LEN: resw 1         ; size of execution block
00000004 ??                                 resb 1
00000005 ??????????             .CPM_CALL:  resb 5         ; ancient call to system
0000000A ????????               .EXIT:              resd 1         ; pointer to exit routine
0000000E ????????               .CTRL_C:    resd 1         ; pointer to ^C routine
00000012 ????????               .FATAL_ABORT:       resd 1         ; pointer to fatal error
00000016 ????                   .PARENT_PID:        resw 1         ; PID of parent (terminate PID)
00000018 <res 14h>              .JFN_TABLE:     resb FILPERPROC ; indices into system table
0000002C ????                   .ENVIRON:   resw 1         ; seg addr of environment
0000002E ????????               .USER_STACK:        resd 1         ; stack of self during system calls
00000032 ????                   .JFN_LENGTH:        resw 1         ; number of handles allowed
00000034 ????????               .JFN_POINTER:       resd 1         ; pointer to JFN table
00000038 ????????               .NEXT_PDB:  resd 1         ; pointer to nested PDB's
0000003C ??                     .INTERCON:  resb 1         ; *** jh-3/28/90 ***
0000003D ??                     .APPEND:    resb 1         ; *** Not sure if still used ***
0000003E ????                   .NOVELL_USED:       resb 2         ; Novell shell (redir) uses these
00000040 ????                   .VERSION:   resw 1         ; DOS version reported to this app
00000042 <res Eh>               .PAD1:              resb 14        ;
00000050 ??????????             .CALL_SYSTEM:       resb 5         ; portable method of system call
00000055 ????????????????       .PAD2:              resb 7         ; reserved so FCB 1 can be used as an extended FCB
0000005C <res 10h>              .FCB1:              resb 16        ; default FCB 1
0000006C <res 10h>              .FCB2:              resb 16        ; default FCB 2
0000007C ????????               .PAD3:              resb 4         ; not sure if this is used by PDB_FCB2
00000080 <res 80h>              .TAIL:              resb 128       ; command tail and default DTA
                                ;.size:
                                endstruc

                                ; --------------------------------------------------------------------------
                                ; <system call definitions>

                                ; 23/03/2019 - Retro DOS v4.0

                                ; (MSDOS 6.0 - SYSCALL.INC, 1991)

                                ABORT                           EQU 0   ;  0      0
                                STD_CON_INPUT                   EQU 1   ;  1      1
                                STD_CON_OUTPUT                  EQU 2   ;  2      2
                                STD_AUX_INPUT                   EQU 3   ;  3      3
                                STD_AUX_OUTPUT                  EQU 4   ;  4      4
                                STD_PRINTER_OUTPUT              EQU 5   ;  5      5
                                RAW_CON_IO                      EQU 6   ;  6      6
                                RAW_CON_INPUT                   EQU 7   ;  7      7
                                STD_CON_INPUT_NO_ECHO           EQU 8   ;  8      8
                                STD_CON_STRING_OUTPUT           EQU 9   ;  9      9
                                STD_CON_STRING_INPUT            EQU 10  ; 10      A
                                STD_CON_INPUT_STATUS            EQU 11  ; 11      B
                                STD_CON_INPUT_FLUSH             EQU 12  ; 12      C
                                DISK_RESET                      EQU 13  ; 13      D
                                SET_DEFAULT_DRIVE               EQU 14  ; 14      E
                                FCB_OPEN                        EQU 15  ; 15      F
                                FCB_CLOSE                       EQU 16  ; 16     10
                                DIR_SEARCH_FIRST                EQU 17  ; 17     11
                                DIR_SEARCH_NEXT                 EQU 18  ; 18     12
                                FCB_DELETE                      EQU 19  ; 19     13
                                FCB_SEQ_READ                    EQU 20  ; 20     14
                                FCB_SEQ_WRITE                   EQU 21  ; 21     15
                                FCB_CREATE                      EQU 22  ; 22     16
                                FCB_RENAME                      EQU 23  ; 23     17
                                GET_DEFAULT_DRIVE               EQU 25  ; 25     19
                                SET_DMA                         EQU 26  ; 26     1A
                                GET_DEFAULT_DPB                 EQU 31  ; 31     1F
                                FCB_RANDOM_READ                 EQU 33  ; 33     21
                                FCB_RANDOM_WRITE                EQU 34  ; 34     22
                                GET_FCB_FILE_LENGTH             EQU 35  ; 35     23
                                GET_FCB_POSITION                EQU 36  ; 36     24
                                SET_INTERRUPT_VECTOR            EQU 37  ; 37     25
                                CREATE_PROCESS_DATA_BLOCK       EQU 38  ; 38     26
                                FCB_RANDOM_READ_BLOCK           EQU 39  ; 39     27
                                FCB_RANDOM_WRITE_BLOCK          EQU 40  ; 40     28
                                PARSE_FILE_DESCRIPTOR           EQU 41  ; 41     29
                                GET_DATE                        EQU 42  ; 42     2A
                                SET_DATE                        EQU 43  ; 43     2B
                                GET_TIME                        EQU 44  ; 44     2C
                                SET_TIME                        EQU 45  ; 45     2D
                                SET_VERIFY_ON_WRITE             EQU 46  ; 46     2E
                                ; Extended functionality group
                                GET_DMA                         EQU 47  ; 47     2F
                                GET_VERSION                     EQU 48  ; 48     30
                                KEEP_PROCESS                    EQU 49  ; 49     31
                                GET_DPB                         EQU 50  ; 50     32
                                SET_CTRL_C_TRAPPING             EQU 51  ; 51     33
                                GET_INDOS_FLAG                  EQU 52  ; 52     34
                                GET_INTERRUPT_VECTOR            EQU 53  ; 53     35
```

```
21693                           GET_DRIVE_FREESPACE             EQU 54  ; 54     36
21694                           CHAR_OPER                       EQU 55  ; 55     37
21695                           INTERNATIONAL                   EQU 56  ; 56     38
21696                           ;   Directory Group
21697                           MKDIR                           EQU 57  ; 57     39
21698                           RMDIR                           EQU 58  ; 58     3A
21699                           CHDIR                           EQU 59  ; 59     3B
21700                           ;   File Group
21701                           CREAT                           EQU 60  ; 60     3C
21702                           OPEN                            EQU 61  ; 61     3D
21703                           CLOSE                           EQU 62  ; 62     3E
21704                           READ                            EQU 63  ; 63     3F
21705                           WRITE                           EQU 64  ; 64     40
21706                           UNLINK                          EQU 65  ; 65     41
21707                           LSEEK                           EQU 66  ; 66     42
21708                           CHMOD                           EQU 67  ; 67     43
21709                           IOCTL                           EQU 68  ; 68     44
21710                           XDUP                            EQU 69  ; 69     45
21711                           XDUP2                           EQU 70  ; 70     46
21712                           CURRENT_DIR                     EQU 71  ; 71     47
21713                           ;    Memory Group
21714                           ALLOC                           EQU 72  ; 72     48
21715                           DEALLOC                         EQU 73  ; 73     49
21716                           SETBLOCK                        EQU 74  ; 74     4A
21717                           ;    Process Group
21718                           EXEC                            EQU 75  ; 75     4B
21719                           EXIT                            EQU 76  ; 76     4C
21720                           WAITPROCESS             EQU 77  ; 77     4D
21721                           FIND_FIRST                      EQU 78  ; 78     4E
21722                           ;   Special Group
21723                           FIND_NEXT                       EQU 79  ; 79     4F
21724                           ; SPECIAL SYSTEM GROUP
21725                           SET_CURRENT_PDB                 EQU 80  ; 80     50
21726                           GET_CURRENT_PDB                 EQU 81  ; 81     51
21727                           GET_IN_VARS                     EQU 82  ; 82     52
21728                           SETDPB                          EQU 83  ; 83     53
21729                           GET_VERIFY_ON_WRITE             EQU 84  ; 84     54
21730                           DUP_PDB                         EQU 85  ; 85     55
21731                           RENAME                          EQU 86  ; 86     56
21732                           FILE_TIMES                      EQU 87  ; 87     57
21733                           ;
21734                           ALLOCOPER               EQU 88  ; 88     58
21735                           ; Network extention system calls
21736                           GetExtendedError        EQU 89  ; 89     59
21737                           CreateTempFile                  EQU 90  ; 90     5A
21738                           CreateNewFile                   EQU 91  ; 91     5B
21739                           LockOper                EQU 92  ; 92     5C Lock and Unlock
21740                           ServerCall              EQU 93  ; 93     5D CommitAll, ServerDOSCall,
21741                                                           ;           CloseByName, CloseUser,
21742                                                           ;           CloseUserProcess,
21743                                                           ;           GetOpenFileList
21744                           UserOper                EQU 94  ; 94     5E Get and Set
21745                           AssignOper              EQU 95  ; 95     5F On, Off, Get, Set, Cancel
21746                           xNameTrans              EQU 96  ; 96     60
21747                           PathParse               EQU 97  ; 97     61
21748                           GetCurrentPSP                   EQU 98  ; 98     62
21749                           Hongeul                 EQU 99  ; 99     63
21750                           ECS_CALL                EQU 99  ; 99     63  ;; DBCS support
21751                           Set_Printer_Flag        EQU 100 ; 100    64
21752                           GetExtCntry             EQU 101 ; 101    65
21753                           GetSetCdPg              EQU 102 ; 102    66
21754                           ExtHandle               EQU 103 ; 103    67
21755                           Commit                          EQU 104 ; 104    68
21756                           GetSetMediaID                   EQU 105 ; 105    69
21757                           IFS_IOCTL               EQU 107 ; 107    6B
21758                           ExtOpen                 EQU 108 ; 108    6C
21759                           ;
21760                           ;ifdef ROMEXEC
21761                           ;ROM_FIND_FIRST                 EQU 109 ; 109     6D
21762                           ;ROM_FIND_NEXT                  EQU 110 ; 110     6E
21763                           ;ROM_EXCLUDE                    EQU 111 ; 111    6F
21764                           ;endif
21765                           ;
21766                           Set_Oem_Handler         EQU 248 ; 248    F8
21767                           OEM_C1                          EQU 249 ; 249    F9
21768                           OEM_C2                          EQU 250 ; 250    FA
21769                           OEM_C3                          EQU 251 ; 251    FB
21770                           OEM_C4                          EQU 252 ; 252    FC
21771                           OEM_C5                          EQU 253 ; 253    FD
21772                           OEM_C6                          EQU 254 ; 254    FE
21773                           OEM_C7                          EQU 255 ; 255    FF
21774
21775                           ; ----------------------------------------------------------------------
21776                           ; SYSCONF.ASM (MSDOS 3.3 - 24/07/1987)
21777                           ; ----------------------------------------------------------------------
21778
21779                           ;;  IF      STACKSW
21780
21781                           ;;
21782                           ;; Internal Stack Parameters
21783                           ;EntrySize          equ     8
21784                           ;
21785                           ;MinCount           equ     8
21786                           ;DefaultCount               equ     9
21787                           ;MaxCount           equ     64
21788                           ;
21789                           ;MinSize            equ     32
21790                           ;DefaultSize                equ     128
21791                           ;MaxSize            equ     512
21792
21793                           ;;  ENDIF
21794
21795                           ; ----------------------------------------------------------------------
21796                           ; BIOSTRUC.INC (MSDOS 3.3 - 24/07/1987)
21797                           ; ----------------------------------------------------------------------
21798                                                           ;;Rev 3.30 Modification
21799                           ; ROM BIOS CALL PACKET STRUCTURES
21800
21801                           ;*********************************
21802                           ;System Service call ( Int 15h )
21803                           ;*********************************
21804                           ;Function AH = 0C0h, Return system configuration
21805                           ;For PC and PCJR on return:
21806                           ;   (AH)    = 80h
21807                           ;   (CY)    = 1
21808                           ;For PCXT, PC PORTABLE and PCAT on return:
21809                           ;   (AH)    = 86h
21810                           ;   (CY)    = 1
21811                           ;For all others:
21812                           ;   (AH)    = 0
21813                           ;   (CY)    = 0
21814                           ;   (ES:BX) = pointer to system descriptor vector in ROS
21815                           ; System descriptor :
21816                           ;   DW      xxxx            length of descriptor in bytes,
```

```
                        ;                       minimum length = 8
                        ;   DB      xx          model byte
                        ;                       0FFh    = PC
                        ;                       0FEh    = PC/XT, Portable
                        ;                       0FDh    = PC/JR
                        ;                       0FCh    = PC/AT
                        ;                       0F9h    = Convertable
                        ;                       0F8h    = Model 80
                        ;                       0E0 thru 0EFh = reserved
                        ;
                        ;   DB      xx          secondary model byte
                        ;                       000h    = PC1
                        ;                       000h    = PC/XT, Portable
                        ;                       000h    = PC/JR
                        ;                       000h    = PC/AT
                        ;                       001h    = PC/AT Model 339
                        ;                       003h    = PC/RT
                        ;                       000h    = Convertable
                        ;
                        ;   DB      xx          bios revision level
                        ;                       00 for first release, subsequent release
                        ;                       of code with same model byte and
                        ;                       secondary model byte require revison level
                        ;                       to increase by one.
                        ;
                        ;   DB      xx          feature information byte 1
                        ;                       X0000000 = 1, bios use DMA channel 3
                        ;                              = 0, DMA channel 3 not used
                        ;
                        ;                       0X000000 = 1, 2nd Interrupt chip present
                        ;                              = 0, 2nd Interrupt chip not present
                        ;
                        ;                       00X00000 = 1, Real Time Clock present
                        ;                              = 0, Real Time Clock not present
                        ;
                        ;                       000X0000 = 1, Keyboard escape sequence(INT 15h)
                        ;                                    called in keyboard interrupt
                        ;                                    (Int 09h).
                        ;                                  = 0, Keyboard escape sequence not
                        ;                                       called.
                        ;                               0000XXXX reserved
                        ;
                        ;   DB      xx          feature information byte 2 - reserved
                        ;
                        ;   DB      xx          feature information byte 2 - reserved
                        ;
                        ;   DB      xx          feature information byte 2 - reserved
                        ;
                        ;   DB      xx          feature information byte 2 - reserved
                        ;
                        ; 22/03/2019
                        struc ROMBIOS_DESC      ; BIOS_SYSTEM_DESCRIPTOR
00000000 ????           .bios_sd_leng:          resw 1
00000002 ??             .bios_sd_modelbyte:     resb 1
                        .bios_sd_scnd_modelbyte:
00000003 ??                                     resb 1
00000004 ??                                     resb 1
00000005 ??             .bios_sd_featurebyte1:  resb 1
00000006 ????????                              resb 4
                        endstruc

                        ;FeatureByte1      bit map equates
                        DMAchannel3         equ 10000000b
                        ScndIntController   equ 01000000b
                        RealTimeClock           equ 00100000b
                        KeyEscapeSeq            equ 00010000b
                                                ;;End of Modification

                        ; ----------------------------------------------------------------------
                        ; SYSVAR.INC (MSDOS 6.0 - 1991)
                        ; ----------------------------------------------------------------------
                        ; 22/03/2019 - Retro DOS v4.0

                        ;   SCCSID = @(#)sysvar.asm 1.1 85/04/10

                        struc SysInitVars
                        ; MSDOS 3.3
00000000 ????????       .SYSI_DPB:    resd 1                    ; DPB chain
00000004 ????????       .SYSI_SFT:    resd 1                    ; SFT chain
00000008 ????????       .SYSI_CLOCK:  resd 1                    ; CLOCK device
0000000C ????????       .SYSI_CON:    resd 1                    ; CON device
00000010 ????           .SYSI_MAXSEC: resw 1                    ; maximum sector size
00000012 ????????       .SYSI_BUF:    resd 1                    ; buffer chain
00000016 ????????       .SYSI_CDS:    resd 1                    ; CDS list
0000001A ????????       .SYSI_FCB:    resd 1                    ; FCB chain
0000001E ????           .SYSI_KEEP:   resw 1                    ; keep count
00000020 ??             .SYSI_NUMIO:  resb 1                    ; number of block devices
00000021 ??             .SYSI_NCDS:   resb 1                    ; number of CDS's
00000022 ????????       .SYSI_DEV:    resd 1                    ; device list
                        ; MSDOS 6.0
00000026 ????           .SYSI_ATTR:     resw 1            ; null device attribute word
00000028 ????           .SYSI_STRAT:          resw 1            ; null device strategy entry point
0000002A ????           .SYSI_INTER:          resw 1            ; null device interrupt entry point
0000002C ????????????   .SYSI_NAME:     resb 8            ; null device name
                        .SYSI_SPLICE:         resb 0            ; TRUE -> splicees being done
00000034 ????           .SYSI_IBMDOS_SIZE:  resw 1        ; DOS size in paragraphs
00000036 ????????       .SYSI_IFS_DOSCALL@: resd 1        ; IFS DOS service rountine entry
0000003A ????????       .SYSI_IFS:      resd 1            ; IFS header chain
0000003E ????????       .SYSI_BUFFERS:        resw 2            ; BUFFERS= values (m,n)
00000042 ??             .SYSI_BOOT_DRIVE:   resb 1        ; boot drive A=1 B=2,..
00000043 ??             .SYSI_DWMOVE:         resb 1            ; 1 if 386 machine
00000044 ????           .SYSI_EXT_MEM:        resw 1            ; Extended memory size in KB.
                        .size:
                        endstruc

                        ;This is added for more information exchage between DOS, BIOS.
                        ;DOS will give the pointer to SysInitTable in ES:DI. - J.K. 5/29/86

                        ; 22/03/2019
                        struc SysInitVars_Ext
00000000 ????????       .SYSI_InitVars:       resd 1      ; Points to the above structure.
00000004 ????????       .SYSI_Country_Tab: resd 1 ; DOS_Country_cdpg_info
                        endstruc

                        ; 09/06/2018
                        ; 08/06/2018 - Retro DOS v3.0 (MSDOS 3.3)
                        SYSI_DPB    equ    0
                        SYSI_SFT    equ 4
                        SYSI_CLOCK  equ 8
                        SYSI_CON    equ 12
                        SYSI_MAXSEC equ 16
                        SYSI_BUF    equ 18
                        SYSI_CDS    equ 22
```

```
                                SYSI_FCB    equ 26
                                SYSI_KEEP   equ 30
                                SYSI_NUMIO  equ    32
                                SYSI_NCDS   equ    33
                                SYSI_DEV    equ 34
                                ; 22/03/2019 - Retro DOS v4.0 (MSDOS 6.0)
                                SYSI_ATTR     equ 38
                                SYSI_STRAT    equ 40
                                SYSI_INTER    equ 42
                                SYSI_NAME     equ 44
                                SYSI_SPLICE   equ 52
                                SYSI_IBMDOS_SIZE  equ 53
                                SYSI_IFS_DOSCALL@ equ 55
                                SYSI_IFS      equ 59
                                SYSI_BUFFERS         equ 63
                                SYSI_BOOT_DRIVE   equ 67
                                SYSI_DWMOVE   equ 68
                                SYSI_EXT_MEM         equ 69

                                ;The SYSI_BUF of SysInitVars points to the following structure

                                EMS_MAP_BUFF_SIZE EQU 12  ; EMS map buffer size

                                struc BUFFINF      ; BUFFINFO
00000000 ????????               .Buff_Queue:          resd 1      ; Head of list of buffers
00000004 ????                   .Dirty_Buff_Count: resw 1 ; number of dirty buffers in list
00000006 ????????               .Cache_ptr:    resd 1      ; pointer to secondary cache
0000000A ????                   .Cache_count:      resw 1 ; number of secondary cache entries

0000000C ??                     .Buff_In_HMA:         resb 1      ; flag to indicate that buffers
                                                          ; are in HMA
0000000D ????????               .Lo_Mem_Buff:         resd 1      ; Ptr to scratch buff in Low Mem
                                                          ;  used to read/write on disks
00000011 ????????               .UU_EMS_FIRST_PAGE:       resw 2
00000015 ????                   .UU_EMS_NPA640:           resw 1
00000017 ??                     .UU_EMS_mode:             resb 1   ; no EMS = -1
00000018 ????                   .UU_EMS_handle:           resw 1   ; EMS handle for buffers
0000001A ????                   .UU_EMS_PageFrame_Number: resw 1 ; EMS page frame number
0000001C ????                   .UU_EMS_Seg_Cnt:   resw 1   ; EMS segment count
0000001E ????                   .UU_EMS_Page_Frame:       resw 1   ; EMS page frame segment address
00000020 ????                   .UU_EMS_reserved:  resw 1   ; EMS segment count
00000022 ??                     .UU_EMS_Map_Buff:  resb 1   ; map buffer
                                .size:
                                endstruc

                                ; ---------------------------------------------------------------------------
                                ; CURDIR.INC (MSDOS 6.0 - 1991)
                                ; ---------------------------------------------------------------------------
                                ; 22/03/2019 - Retro DOS v4.0

                                ;** CDS - Current Directory Structure
                                ;
                                ; CDS items are used bu the internal routines to store cluster numbers and
                                ; network identifiers for each logical name.  The ID field is used dually,
                                ; both as net ID and for a cluster number for local devices. In the case
                                ; of local devices, the cluster number will be -1 if there is a potential
                                ; of the disk being changed or if the path must be recracked.
                                ;
                                ;   Some pathnames have special preambles, such as
                                ;
                                ;          \\machine\sharename\...
                                ;   For these pathnames we can't allow ".." processing to back us
                                ;   up into the special front part of the name.  The CURDIR_END field
                                ;   holds the address of the seperator character which marks
                                ;   the split between the special preamble and the regular
                                ;   path list; ".." processing isn't allowed to back us up past
                                ;   (i.e., before) CURDIR_END
                                ;   For the root, it points at the leading /. For net
                                ;   assignments it points at the end (nul) of the initial assignment:
                                ;   A:/    \\foo\bar          \\foo\bar\blech\bozo
                                ;     ^            ^                  ^

                                DIRSTRLEN   EQU    64+3          ; Max length in bytes of directory strings
                                TEMPLEN     EQU    DIRSTRLEN*2

                                struc              curdir_list
                                ; MSDOS 3.3
00000000 <res 43h>              .cdir_text  resb   DIRSTRLEN     ; text of assignment and curdir
00000043 ????                   .cdir_flags resw   1             ; various flags
00000045 ????????               .cdir_devptr       resd   1            ; local pointer to DPB or net device
00000049 ????????               .cdir_ID    resw   2             ; cluster of current dir (net ID)
0000004D ????                   .cdir_usr_word     resw   1
0000004F ????                   .cdir_end   resw   1             ; end of assignment
                                ; MSDOS 6.0
00000051 ??                     .cdir_type: resb   1             ; IFS drive (2=ifs, 4=netuse)
00000052 ????????               .cdir_ifd_hdr:     resd   1            ; Ptr to File System Header
00000056 ????                   .cdir_fsda: resb   2             ; File System Dependent Data Area
                                .size:
                                endstruc

                                curdirlen   EQU    curdir_list.size      ; Needed for screwed up
                                                                         ; ASM87 which doesn't allow
                                                                         ; Size directive as a macro
                                                                         ; argument
                                %define curdir_netID      dword [curdir_list.cdir_ID]

                                ;** Flag values for CURDIR_FLAGS

                                ;Flag word masks
                                curdir_isnet       EQU    1000000000000000B
                                curdir_isifs       EQU    1000000000000000B
                                curdir_inuse       EQU    0100000000000000B
                                curdir_splice      EQU    0010000000000000B
                                curdir_local       EQU    0001000000000000B

                                ; ---------------------------------------------------------------------------
                                ; SF.INC (MSDOS 6.0 - 1991)
                                ; ---------------------------------------------------------------------------
                                ; 25/03/2019 - Retro DOS v4.0

                                ; 09/04/2024 - Retro DOS v4.2 (BugFix)
                                ; 09/04/2024 - Retro DOS v5.0

                                ; system file table

                                ;** System File Table SuperStructure
                                ;
                                ;   The system file table entries are allocated in contiguous groups.
                                ;   There may be more than one such groups; the SF "superstructure"
                                ;   tracks the groups.

                                struc      SF
00000000 ????????               .SFLink:   resd   1
00000004 ????                   .SFCount:  resw   1              ; number of entries
```

```
22065 00000006 ????                    .SFTable:   resw    1             ; beginning of array of the following
22066                                  .size:
22067                                  endstruc
22068
22069                                  ;** System file table entry
22070                                  ;
22071                                  ;   These are the structures which are at SFTABLE in the SF structure.
22072
22073                                  struc       SF_ENTRY
22074 00000000 ????                    .sf_ref_count:     resw    1              ; number of processes sharing entry
22075                                                                        ;   if FCB then ref count
22076 00000002 ????                    .sf_mode:   resw    1             ; mode of access or high bit on if FCB
22077 00000004 ??                      .sf_attr:   resb    1             ; attribute of file
22078 00000005 ????                    .sf_flags:  resw    1             ;Bits 8-15
22079                                                                       ; Bit 15 = 1 if remote file
22080                                                                       ;       = 0 if local file or device
22081                                                                       ; Bit 14 = 1 if date/time is not to be
22082                                                                       ;   set from clock at CLOSE.  Set by
22083                                                                       ;   FILETIMES and FCB_CLOSE.  Reset by
22084                                                                       ;   other reseters of the dirty bit
22085                                                                       ;   (WRITE)
22086                                                                       ; Bit 13 = Pipe bit (reserved)
22087                                                                       ;
22088                                                                       ; Bits 0-7 (old FCB_devid bits)
22089                                                                       ; If remote file or local file, bit
22090                                                                       ; 6=0 if dirty Device ID number, bits
22091                                                                       ; 0-5 if local file.
22092                                                                       ; bit 7=0 for local file, bit 7
22093                                                                       ;     =1 for local I/O device
22094                                                                       ; If local I/O device, bit 6=0 if EOF (input)
22095                                                                       ;             Bit 5=1 if Raw mode
22096                                                                       ;             Bit 0=1 if console input device
22097                                                                       ;             Bit 1=1 if console output device
22098                                                                       ;             Bit 2=1 if null device
22099                                                                       ;             Bit 3=1 if clock device
22100 00000007 ????????                .sf_devptr: resd    1             ; Points to DPB if local file, points
22101                                                                       ; to device header if local device,
22102                                                                       ; points to net device header if
22103                                                                       ; remote
22104 0000000B ????                    .sf_firclus:       resw    1              ; First cluster of file (bit 15 = 0)
22105                                  ;.sf_lstclus:      resw    1 ; *
22106 0000000D ????                    .sf_time:   resw    1             ; Time associated with file
22107 0000000F ????                    .sf_date:   resw    1             ; Date associated with file
22108 00000011 ????????                .sf_size:   resd    1             ; Size associated with file
22109 00000015 ????????                .sf_position:      resd    1              ; Read/Write pointer or LRU count for FCBs
22110                                  ;
22111                                  ; Starting here, the next 7 bytes may be used by the file system to store an
22112                                  ; ID
22113                                  ;
22114 00000019 ????                    .sf_cluspos:       resw    1              ; Position of last cluster accessed
22115 0000001B ????????                .sf_dirsec: resd    1 ; 09/04/2024 ; Sector number of directory sector for this file
22116 0000001F ??                      .sf_dirpos: resb    1             ; Offset of this entry in the above
22117                                  ;
22118                                  ; End of 7 bytes of file-system specific info.
22119                                  ;
22120 00000020 <res Bh>                .sf_name:   resb    11            ; 11 character name that is in the
22121                                                                       ; directory entry.  This is used by
22122                                                                       ; close to detect file deleted and
22123                                                                       ; disk changed errors.
22124                                  ; SHARING INFO
22125 0000002B ????????                .sf_chain:  resd    1             ; link to next SF
22126 0000002F ????                    .sf_UID:    resw    1
22127 00000031 ????                    .sf_PID:    resw    1
22128 00000033 ????                    .sf_MFT:    resw    1
22129 00000035 ????                    .sf_lstclus:       resw    1 ; *          ; Last cluster accessed
22130 00000037 ????????                .sf_IFS_HDR:       resd    1 ; **
22131                                  .size:
22132                                  endstruc
22133
22134                                  ; --------------------------------------------------------------------------
22135                                  ; DOSCNTRY.INC (MSDOS 3.3 - 24/07/1987)
22136                                  ; --------------------------------------------------------------------------
22137                                  ; 11/06/2018 - Retro DOS v3.0
22138
22139                                  ;Equates for COUNTRY INFORMATION.
22140                                  SetCountryInfo             EQU     1       ;country info
22141                                  SetUcase           EQU     2       ;uppercase table
22142                                  SetLcase           EQU     3       ;lowercase table (Reserved)
22143                                  SetUcaseFile               EQU     4       ;uppercase file spec table
22144                                  SetFileList        EQU     5       ;valid file character list
22145                                  SetCollate         EQU     6       ;collating sequence
22146                                  SetDBCS            EQU     7       ;double byte character set
22147                                  SetALL                     EQU     -1      ;all the entries
22148
22149                                  ;DOS country and code page information table structure.
22150                                  ;Internally, IBMDOS gives a pointer to this table.
22151                                  ;IBMBIO, MODE and NLSFUNC modules communicate with IBMDOS through
22152                                  ;this structure.
22153
22154                                  struc country_cdpg_info ; DOS_country_cdpg_info
22155 00000000 ??????????????????      .ccInfo_reserved : resb    8       ;reserved for internal use
22156 00000008 <res 40h>               .ccPath_CountrySys:        resb    64      ;path and filename for country info
22157 00000048 ????                    .ccSysCodePage:            resw    1       ;system code page id
22158 0000004A ????                    .ccNumber_of_entries:      resw    1 ; dw 5
22159 0000004C ??                      .ccSetUcase:               resb    1 ; db SetUcase ; = 2
22160 0000004D ????????                .ccUcase_ptr:              resd    1       ;pointer to Ucase table
22161
22162 00000051 ??                      .ccSetUcaseFile:   resb    1 ; db SetUcaseFile ; = 4
22163 00000052 ????????                .ccFileUcase_ptr:  resd    1       ;pointer to File Ucase table
22164
22165 00000056 ??                      .ccSetFileList:            resb    1 ; db SetFileList ; = 5
22166 00000057 ????????                .ccFileChar_ptr:   resd    1       ;pointer to File char list table
22167
22168 0000005B ??                      .ccSetCollate:             resb    1 ; db SetCollate ; = 6
22169 0000005C ????????                .ccCollate_ptr:            resd    1       ;pointer to collate table
22170
22171 00000060 ??                      .ccSetCountryInfo: resb    1 ; db SetCountryInfo ; = 1
22172 00000061 ????                    .ccCountryInfoLen: resw    1       ;length of country info
22173 00000063 ????                    .ccDosCountry:             resw    1       ;system country code id
22174 00000065 ????                    .ccDosCodePage:            resw    1       ;system code page id
22175 00000067 ????                    .ccDFormat:        resw    1       ;date format
22176 00000069 ??????????              .ccCurSymbol:              resb    5 ; db "    ",0
22177                                                                       ;5 byte of (currency symbol+0)
22178 0000006E ????                    .cc1000Sep:        resb    2 ; db " ",0 ;2 byte of (1000 sep. + 0)
22179 00000070 ????                    .ccDecSep:         resb    2 ; db " ",0 ;2 byte of (Decimal sep. + 0)
22180 00000072 ????                    .ccDateSep:        resb    2 ; db " ",0 ;2 byte of (date sep. + 0)
22181 00000074 ????                    .ccTimeSep:        resb    2 ; db " ",0 ;2 byte of (time sep. + 0)
22182 00000076 ??                      .ccCFormat:        resb    1       ;currency format flags
22183 00000077 ??                      .ccCSigDigits:             resb    1       ;# of digits in currency
22184 00000078 ??                      .ccTFormat:        resb    1       ;time format
22185 00000079 ????????                .ccMono_Ptr:               resd    1       ;monocase routine entry point
22186 0000007D ????                    .ccListSep:        resb    2 ; db " ",0 ;data list separator
22187 0000007F <res Ah>                .ccReserved_area:  resw    5 ; dw 5 dup(?) ;reserved
22188                                  .size:
```

```
22189                                   endstruc
22190
22191                                   NEW_COUNTRY_SIZE    equ  country_cdpg_info.size - country_cdpg_info.ccDosCountry
22192
22193                                   ; ==========================================================================
22194                                   ; retrodos4.s (offset addresses in MSDOS.SYS or RETRODOS.SYS)
22195                                   ; ==========================================================================
22196                                   ; 21/03/2019 - Retro DOS v4.0
22197                                   ; 21/10/2022 - Retro DOS v4.0 (MOdified MSDOS 5.0 IO.SYS)
22198
22199                                   ;KERNEL_SEGMENT    equ 0070h  ; (IO.SYS loading segment, BIOS_DATA segment)
22200                                   ; 21/10/2022
22201                                   DOSBIODATASEG equ 0070h   ; (IO.SYS loading segment, BIOS_DATA segment)
22202                                   ; 22/10/2022
22203                                   ;DOSBIOCODESEG equ 02C7h ; (MSDOS 5.0 IO.SYS, BIOS_CODE segment)
22204                                   ; 09/12/2022
22205                                   DOSBIOCODESEG equ IOSYSCODESEG
22206
22207                                   ; Note: These offset addresses must be changed when the code
22208                                   ;   in retrodos4.s (MSDOS.SYS) file will be changed.
22209
22210                                   ; (following addresses can be verified by searching them in retrodos4.lst)
22211
22212                                   ; 09/12/2022
22213                                   %if 0
22214
22215                                   ; 13/05/2019
22216
22217                                   ;IsWin386         equ 08CFh
22218                                   ;V86_Crit_SetFocus equ 08D0h
22219                                   ; 21/10/2022
22220                                   IsWin386          equ 08D0h
22221                                   V86_Crit_SetFocus equ 08D1h
22222
22223                                   ;seg_reinit  equ 0772h ; not used in Retro DOS v4.0
22224                                   ; 21/10/2022 - Retro DOS v4.0 (MOdified MSDOS 5.0 IO.SYS)
22225                                   seg_reinit    equ 0032h ; DOSBIOCODESEG:0032h
22226
22227                                   ;SysinitPresent      equ 08FCh
22228                                   ; 21/10/2022
22229                                   SysinitPresent       equ 08FDh
22230
22231                                   inHMA                equ 000Dh
22232                                   xms           equ 000Eh
22233                                   ;FreeHMAPtr   equ 08F6h
22234                                   ;multrk_flag         equ 0533h
22235                                   ;ec35_flag    equ 0535h
22236                                   ;EOT          equ 012Eh
22237                                   ; 21/10/2022
22238                                   FreeHMAPtr    equ 08F7h
22239                                   multrk_flag   equ 052Fh
22240                                   ec35_flag     equ 0531h
22241                                   EOT           equ 012Ch
22242
22243                                   ;NextStack    equ 08BFh
22244                                   ;IT_StackLoc         equ 08C5h
22245                                   ;IT_StackSize        equ 08C9h
22246                                   ; 21/10/2022
22247                                   NextStack     equ 08C0h
22248                                   IT_StackLoc   equ 08C6h
22249                                   IT_StackSize         equ 08CAh
22250
22251                                   ;MoveDOSIntoHMA      equ 08F8h
22252                                   ; 21/10/2022
22253                                   MoveDOSIntoHMA       equ 08F9h
22254
22255                                   ;INT19SEM equ 0644h ; 01/05/2019 - retrodos4.lst
22256                                   ;I19_LST  equ 0645h ; 27/03/2019 - retrodos4.lst
22257                                   ; 21/10/2022
22258                                   INT19SEM equ 0640h ; (iosys5.txt)
22259                                   I19_LST  equ 0641h ; (iosys5.txt)
22260
22261                                   %endif
22262
22263                                   ; 09/12/2022
22264                                   seg_reinit equ _seg_reinit
22265                                   ec35_flag  equ ec35flag
22266                                   INT19SEM   equ int19sem
22267                                   I19_LST    equ i19_lst
22268
22269                                   INT19OLD02 equ I19_LST+1 ; 0642h ; 21/10/2022
22270                                   INT19OLD08 equ I19_LST+6
22271                                   INT19OLD09 equ I19_LST+11
22272                                   INT19OLD0A equ I19_LST+16
22273                                   INT19OLD0B equ I19_LST+21
22274                                   INT19OLD0C equ I19_LST+26
22275                                   INT19OLD0D equ I19_LST+31
22276                                   INT19OLD0E equ I19_LST+36
22277                                   INT19OLD70 equ I19_LST+41
22278                                   INT19OLD72 equ I19_LST+46
22279                                   INT19OLD73 equ I19_LST+51
22280                                   INT19OLD74 equ I19_LST+56
22281                                   INT19OLD76 equ I19_LST+61
22282                                   INT19OLD77 equ I19_LST+66 ; 0683h ; 21/10/2022
22283
22284                                   ; 09/12/2022
22285                                   %if 0
22286
22287                                   ;keyrd_func equ 04E9h
22288                                   ;keysts_func       equ 04EAh
22289                                   ;t_switch   equ 04F6h
22290                                   ; 21/10/2022
22291                                   keyrd_func  equ 04E5h
22292                                   keysts_func equ 04E6h
22293                                   t_switch    equ 04F2h
22294
22295                                   ; 22/10/2022
22296                                   SYSINITSEG  equ 046Dh  ; SYSINIT segment
22297                                   BCODE_END   equ (SYSINITSEG-DOSBIOCODESEG)*16 ; = 1A60h
22298                                   BCODE_START equ 30h  ; (offset BiosDataWord in DOSBIOCODESEG)
22299                                   RE_INIT            equ 089Bh ; (re_init offset in DOSBIODATASEG)
22300
22301                                   %endif
22302
22303                                   ; 09/12/2022
22304                                   BCODESTART  equ BIOSDATAWORD
22305                                   RE_INIT            equ re_init
22306
22307                                   ; ---------------------------------------------------------------------------
22308                                   ; CONFIG.INC (MSDOS 6.0 - 1991)
22309                                   ; ---------------------------------------------------------------------------
22310                                   ; 15/04/2019 - Retro DOS v4.0
22311
22312                                   CONFIG_BEGIN        equ  '['
```

```
                        CONFIG_BREAK        equ  'C'
                        CONFIG_BUFFERS      equ  'B'
                        CONFIG_COMMENT      equ  'Y'
                        CONFIG_COUNTRY      equ  'Q'
                        CONFIG_DEVICE       equ  'D'
                        CONFIG_DEVICEHIGH   equ  'U'
                        CONFIG_DOS          equ  'H'
                        CONFIG_DRIVPARM     equ  'P'
                        CONFIG_FCBS         equ  'X'
                        CONFIG_FILES        equ  'F'
                        CONFIG_INCLUDE      equ  'J'
                        CONFIG_INSTALL      equ  'I'
                        CONFIG_INSTALLHIGH  equ  'W'
                        CONFIG_LASTDRIVE    equ  'L'
                        CONFIG_MENUCOLOR    equ  'R'
                        CONFIG_MENUDEFAULT  equ  'A'
                        CONFIG_MENUITEM     equ  'E'
                        CONFIG_MULTITRACK   equ  'M'
                        CONFIG_NUMLOCK      equ  'N'
                        CONFIG_REM          equ  'O'
                        CONFIG_SEMICOLON    equ  ';'
                        CONFIG_SET          equ  'V'
                        CONFIG_SHELL        equ  'S'
                        CONFIG_STACKS       equ  'K'
                        CONFIG_SUBMENU      equ  'O'
                        CONFIG_SWITCHES     equ  '1'

                        CONFIG_UNKNOWN      equ  'Z'

                        ; 13/05/2024 - Retro DOS v5.0 (PCDOS 71 IBMBIO.COM)
                        CONFIG_DOSDATA      equ  'T'

                        CONFIG_OPTION_QUERY equ 80h

                        ; ----------------------------------------------------------------------
                        ; SYSINIT1.ASM (MSDOS 6.0 - 1991)
                        ; ----------------------------------------------------------------------
                        ; 21/03/2019 - Retro DOS v4.0

                        trueequ     0FFFFh
                        false       equ     0
                        cr  equ     13
                        lf  equ     10
                        tab equ     9

                        multMULT      equ 4Ah
                        multMULTGETHMAPTR  equ     1
                        multMULTALLOCHMA   equ     2

                        ;NOEXEC    equ     FALSE

                        stacksw    equ     true    ;include switchable hardware stacks
                        mycds_size equ     88      ;size of curdir_list. if it is not
                                           ;the same, then will generate compile error.

                        entrysize   equ     8

                        mincount    equ     8
                        defaultcount equ    9
                        maxcount    equ     64

                        minsize     equ     32
                        defaultsize equ     128
                        maxsize     equ     512

                        ;%define allocbyte  byte [es:bp+0]
                        ;%define intlevel   byte [es:bp+1]
                        ;%define savedsp    word [es:bp+2]
                        ;%define savedss    word [es:bp+4]
                        ;%define newsp         word [es:bp+6]

                        allocbyte   equ     0
                        intlevel    equ     1
                        savedsp     equ     2
                        savedss     equ     4
                        newsp       equ     6

                        free    equ     0
                        allocated   equ     1
                        overflowed  equ     2
                        clobbered   equ     3

                        ;---------------------------------------
                        ; external variable defined in ibmbio module for multi-track

                        multrk_on equ      10000000b ;user specified mutitrack=on,or system turns
                                             ; it on after handling config.sys file as a
                                             ; default value,if multrk_flag = multrk_off1.
                        multrk_off1 equ 00000000b ;initial value. no "multitrack=" command entered.
                        multrk_off2 equ 00000001b ;user specified multitrack=off.

                        ; SYSINITSEG       SEGMENT PUBLIC 'SYSTEM_INIT'

                        SYSINIT$:
                            ;IF    STACKSW
                            ; include MSSTACK.INC ;Main stack program and data definitions
                            ; include STKMES.INC  ;Fatal stack error message
                            ;   public Endstackcode
                        ;Endstackcode      label byte
                            ;ENDIF

                        ; 05/07/2018
                        ; ----------------------------------------------------------------------
                        ; 04/06/2018 - Retro DOS v3.0

                        ; ----------------------------------------------------------------------
                        ; 21/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS - SYSINIT)
                        ; ----------------------------------------------------------------------

                        ;   MSStack.inc
                        ;
                        ;   Interrupt level 2, 3, 4, 5, 6, 7,(10, 11, 12, 14, 15 - AT level)
                        ;   should follow the standard Interrupt Sharing Scheme which has
                        ;   a standard header structure.
                        ;   Fyi, the following shows the relations between
                        ;   the interrupt vector and interrupt level.
                        ; VEC(Hex)    2  8  9  A  B  C    D  E  70  72  73  74  76  77
                        ; LVL(Deci)   9  0  1  2  3  4    5  6   8  10  11  12  14  15
                        ;   MSSTACK module modifies the following interrupt vectors
                        ;   to meet the standard Interrupt Sharing standard;
                        ;     A, B, C, D, E, 72, 73, 74, 76, 77.
                        ;   Also, for interrupt level 7 and 15, the FirstFlag in a standard header
                        ;   should be initialized to indicat whether this interrupt handler is
                        ;   the first (= 80h) or not.  The FirstFlag entry of INT77h's
```

```
22437                                   ;   program header is initialized in STKINIT.INC module.
22438                                   ;   FirstFlag is only meaningful for interrupt level 7 and 15.
22439                                   ;
22440                                   
22441                                   ;  User specifies the number of stack elements - default = 9
22442                                   ;                                               minimum = 8
22443                                   ;                                               maximum = 64
22444                                   ;
22445                                   ;  Intercepts Asynchronous Hardware Interrupts only
22446                                   ;
22447                                   ;  Picks a stack from pool of stacks and switches to it
22448                                   ;
22449                                   ;  Calls the previously saved interrupt vector after pushing flags
22450                                   ;
22451                                   ;  On return, returns the stack to the stack pool
22452                                   ;
22453                                   
22454                                   ; This is a modification of STACKS:
22455                                   ; 1. To fix a bug which was causing the program to take up too much space.
22456                                   ; 2. To dispense stack space from hi-mem first rather than low-mem first.
22457                                   ;    . Clobbers the stack that got too big instead of innocent stack
22458                                   ;    . Allows system to work if the only stack that got too big was the most
22459                                   ;      deeply nested one
22460                                   ; 3. Disables NMI interrupts while setting the NMI vector.
22461                                   ; 4. Does not intercept any interrupts on a PCjr.
22462                                   ; 5. Double checks that a nested interrupt didn't get the same stack.
22463                                   ; 6. Intercepts Ints 70, 72-77 for PC-ATs and other future products
22464                                   
22465                                               ;EVEN
22466                                   ;align 2
22467                                               ; 21/10/2022
22468                                   
22469 00000000 0000                               dw      0       ; spare field but leave these in order
22470 00000002 0000                   stackcount: dw      0
22471 00000004 0000                   stackat:    dw      0
22472 00000006 0000                   stacksize:  dw      0
22473 00000008 0000                   stacks:             dw      0
22474 0000000A 0000                               dw      0
22475                                   
22476 0000000C [0800]                 firstentry: dw      stacks
22477 0000000E [4800]                 lastentry:  dw      stacks+(defaultcount*entrysize)-entrysize
22478 00000010 [4800]                 nextentry:  dw      stacks+(defaultcount*entrysize)-entrysize
22479                                   
22480                                   ;******************************************************************************
22481                                   ; THESE ARE THE INDIVIDUAL INTERRUPT HANDLERS
22482                                   
22483                                   ; ---------------------------------------------------------------------------
22484                                   
22485 00000012 00000000               old02:      dd      0
22486                                   
22487                                   int02:
22488                                   
22489                                   ; ****************************************************************************
22490                                   ;
22491                                   ; this is special support for the pc convertible / nmi handler
22492                                   ;
22493                                   ;   on the pc convertible, there is a situation where an nmi can be
22494                                   ;   caused by using the "out" instructions to certain ports. when this
22495                                   ;   occurs, the pc convertible hardware *guarantees* that **nothing**
22496                                   ;   can stop the nmi or interfere with getting to the nmi handler. this
22497                                   ;   includes other type of interrupts (hardware and software), and
22498                                   ;   also includes other type of nmi's. when any nmi has occured,
22499                                   ;   no other interrtupt (hardware, software or nmi) can occur until
22500                                   ;   the software takes specific steps to allow further interrupting.
22501                                   ;
22502                                   ;   for pc convertible, the situation where the nmi is generated by the
22503                                   ;   "out" to a control port requires "fixing-up" and re-attempting. in
22504                                   ;   otherwords, it is actually a "restartable exception". in this
22505                                   ;   case, the software handler must be able to get to the stack in
22506                                   ;   order to figure out what instruction caused the problem, where
22507                                   ;   it was "out"ing to and what value it was "out"ing.  therefore,
22508                                   ;   we will not switch stacks in this situation. this situation is
22509                                   ;   detected by interrogating port 62h, and checking for a bit value
22510                                   ;   of 80h. if set, *****do not switch stacks*****.
22511                                   ;
22512                                   ; ****************************************************************************
22513                                   
22514 00000016 50                           push    ax
22515 00000017 06                           push    es
22516 00000018 B800F0                       mov     ax,0F000h
22517 0000001B 8EC0                         mov     es,ax
22518                                       ; 02/11/2022
22519 0000001D 26803EFEFFF9                 cmp     byte [es:0FFFEh],0F9h ; mdl_convert ; check if convertible
22520 00000023 07                           pop     es
22521 00000024 750C                         jne     short normal02
22522                                   
22523 00000026 E462                         in      al,62h          ; PC/XT PPI port C. Bits:
22524                                                               ; 0-3: values of DIP switches
22525                                                               ; 5: 1=Timer 2 channel out
22526                                                               ; 6: 1=I/O channel check
22527                                                               ; 7: 1=RAM parity check error occurred.
22528 00000028 A880                         test    al,80h
22529 0000002A 7406                         jz      short normal02
22530                                   special02:
22531 0000002C 58                           pop     ax
22532 0000002D 2EFF2E[1200]                 jmp     far [cs:old02]
22533                                   normal02:
22534 00000032 58                           pop     ax
22535 00000033 E81101                       call    do_int_stacks
22536 00000036 [1200]                       dw      old02
22537                                   
22538                                   ; ---------------------------------------------------------------------------
22539                                   
22540 00000038 00000000               old08:      dd      0
22541                                   
22542                                   int08:
22543 0000003C E80801                       call    do_int_stacks
22544 0000003F [3800]                       dw      old08
22545                                   
22546                                   ; ---------------------------------------------------------------------------
22547                                   
22548 00000041 00000000               old09:      dd      0
22549                                   
22550                                   int09:
22551                                   
22552                                   ; keyboard interrupt must have a three byte jump, a nop and a zero byte
22553                                   ; as its first instruction for compatibility reasons
22554                                   
22555 00000045 EB02                         jmp     short keyboard_lbl
22556 00000047 90                           nop
22557 00000048 00                           db      0
22558                                   
22559                                   keyboard_lbl:
22560 00000049 E8FB00                       call    do_int_stacks
```

```
                                    dw      old09
                                ; --------------------------------------------------------------------------
00000004E 00000000              old70:      dd      0

                                int70:
00000052 E8F200                     call    do_int_stacks
00000055 [4E00]                     dw      old70
                                ; --------------------------------------------------------------------------
                                ;   irp     a,<0a,0b,0c,0d,0e,72,73,74,76,77>
                                ;public     int&a
                                ;public     old&a
                                ;public     firstflag&a
                                ;int&a      proc    far
                                ;   jmp     short entry_int&a&_stk
                                ;old&a      dd        0             ;forward pointer
                                ;   dw       424bh          ;compatible signature for int. sharing
                                ;firstflag&a db   0               ;the firstly hooked.
                                ;   jmp     short intret_&a        ;reset routine. we don't care this.
                                ;   db      7 dup (0)      ;reserved for future.
                                ;entry_int&a&_stk:
                                ;   call    do_int_stacks
                                ;   dw      old&a
                                ;intret_&a:
                                ;   iret
                                ;int&a      endp
                                ;   endm
                                ; --------------------------------------------------------------------------
                                int0A:
00000057 EB10                       jmp     short entry_int0A_stk
00000059 00000000               old0A:      dd      0
0000005D 4B42                       dw      424Bh
0000005F 00                     firstflag0A:
                                    db      0
00000060 EB0C                       jmp     short intret_0A
00000062 00<rep 7h>                 times   7 db 0

                                entry_int0A_stk:
00000069 E8DB00                     call    do_int_stacks
0000006C [5900]                     dw      old0A
                                intret_0A:
0000006E CF                         iret

                                ; --------------------------------------------------------------------------

                                int0B:
0000006F EB10                       jmp     short entry_int0B_stk
00000071 00000000               old0B:      dd      0
00000075 4B42                       dw      424Bh
00000077 00                     firstflag0B:
                                    db      0
00000078 EB0C                       jmp     short intret_0B
0000007A 00<rep 7h>                 times   7 db 0

                                entry_int0B_stk:
00000081 E8C300                     call    do_int_stacks
00000084 [7100]                     dw      old0B
                                intret_0B:
00000086 CF                         iret

                                ; --------------------------------------------------------------------------

                                int0C:
00000087 EB10                       jmp     short entry_int0C_stk
00000089 00000000               old0C:      dd      0
0000008D 4B42                       dw      424Bh
0000008F 00                     firstflag0C:
                                    db      0
00000090 EB0C                       jmp     short intret_0C
00000092 00<rep 7h>                 times   7 db 0

                                entry_int0C_stk:
00000099 E8AB00                     call    do_int_stacks
0000009C [8900]                     dw      old0C
                                intret_0C:
0000009E CF                         iret

                                ; --------------------------------------------------------------------------

                                int0D:
0000009F EB10                       jmp     short entry_int0D_stk
000000A1 00000000               old0D:      dd      0
000000A5 4B42                       dw      424Bh
000000A7 00                     firstflag0D:
                                    db      0
000000A8 EB0C                       jmp     short intret_0D
000000AA 00<rep 7h>                 times   7 db 0

                                entry_int0D_stk:
000000B1 E89300                     call    do_int_stacks
000000B4 [A100]                     dw      old0D
                                intret_0D:
000000B6 CF                         iret

                                ; --------------------------------------------------------------------------

                                int0E:
000000B7 EB10                       jmp     short entry_int0E_stk
000000B9 00000000               old0E:      dd      0
000000BD 4B42                       dw      424Bh
000000BF 00                     firstflag0E:
                                    db      0
000000C0 EB0C                       jmp     short intret_0E
000000C2 00<rep 7h>                 times   7 db 0

                                entry_int0E_stk:
000000C9 E87B00                     call    do_int_stacks
000000CC [B900]                     dw      old0E
                                intret_0E:
000000CE CF                         iret

                                ; --------------------------------------------------------------------------

                                int72:
000000CF EB10                       jmp     short entry_int72_stk
000000D1 00000000               old72:      dd      0
000000D5 4B42                       dw      424Bh
                                firstflag72:
000000D7 00                         db      0
```

```
000000D8 EB0C                       jmp     short intret_72
000000DA 00<rep 7h>                 times   7 db 0

                                entry_int72_stk:
000000E1 E86300                     call    do_int_stacks
000000E4 [D100]                     dw      old72
                                intret_72:
000000E6 CF                         iret

                                ; --------------------------------------------------------------------------

                                int73:
000000E7 EB10                       jmp     short entry_int73_stk
000000E9 00000000               old73:      dd      0
000000ED 4B42                       dw      424Bh
                                firstflag73:
000000EF 00                         db      0
000000F0 EB0C                       jmp     short intret_73
000000F2 00<rep 7h>                 times   7 db 0

                                entry_int73_stk:
000000F9 E84B00                     call    do_int_stacks
000000FC [E900]                     dw      old73
                                intret_73:
000000FE CF                         iret

                                ; --------------------------------------------------------------------------

                                int74:
000000FF EB10                       jmp     short entry_int74_stk
00000101 00000000               old74:      dd      0
00000105 4B42                       dw      424Bh
                                firstflag74:
00000107 00                         db      0
00000108 EB0C                       jmp     short intret_74
0000010A 00<rep 7h>                 times   7 db 0

                                entry_int74_stk:
00000111 E83300                     call    do_int_stacks
00000114 [0101]                     dw      old74
                                intret_74:
00000116 CF                         iret

                                ; --------------------------------------------------------------------------

                                int76:
00000117 EB10                       jmp     short entry_int76_stk
00000119 00000000               old76:      dd      0
0000011D 4B42                       dw      424Bh
                                firstflag76:
0000011F 00                         db      0
00000120 EB0C                       jmp     short intret_76
00000122 00<rep 7h>                 times   7 db 0

                                entry_int76_stk:
00000129 E81B00                     call    do_int_stacks
0000012C [1901]                     dw      old76
                                intret_76:
0000012E CF                         iret

                                ; --------------------------------------------------------------------------

                                int77:
0000012F EB10                       jmp     short entry_int77_stk
00000131 00000000               old77:      dd      0
00000135 4B42                       dw      424Bh
                                firstflag77:
00000137 00                         db      0
00000138 EB0C                       jmp     short intret_77
0000013A 00<rep 7h>                 times   7 db 0

                                entry_int77_stk:
00000141 E80300                     call    do_int_stacks
00000144 [3101]                     dw      old77
                                intret_77:
00000146 CF                         iret

                                ; --------------------------------------------------------------------------

                                ;****************************************************************************
                                ;common routines
                                ;****************************************************************************

                                ; do interrupt stack switching. the fake return address holds
                                ; a pointer to the far-pointer of the actual interrupt
                                ; service routine

                                ; 21/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 SYSINIT)
                                ; 21/03/2019 - Retro DOS v4.0

                                ;allocbyte   equ 0
                                ;intlevel    equ 1
                                ;savedsp     equ 2
                                ;savedss     equ 4
                                ;newsp       equ 6

                                ; 30/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 SYSINIT)
                                ; (MSDOS 6.21 IO.SYS, SYSINIT:0147h)

                                do_int_stacks:
00000147 50                         push    ax
00000148 55                         push    bp
00000149 06                         push    es
0000014A 2E8E06[0A00]               mov     es,[cs:stacks+2]      ; Get segment of stacks
0000014F 2E8B2E[1000]               mov     bp,[cs:nextentry]     ; get most likely candidate
00000154 B001                       mov     al,allocated ; 1
                                    ; 21/10/2022
                                    ;xchg   [es:bp+allocbyte],al
                                    ; 11/12/2022
00000156 26864600                   xchg    [es:bp],al            ; grab the entry
0000015A 3C00                       cmp     al,free ; 0           ; still avail?
0000015C 7551                       jne     short notfree02

0000015E 2E832E[1000]08             sub     word [cs:nextentry],entrysize ; set for next interrupt

                                found02:
00000164 26896602                   mov     [es:bp+savedsp],sp    ; save sp value
00000168 268C5604                   mov     [es:bp+savedss],ss    ; save ss also

0000016C 89E8                       mov     ax,bp                 ; temp save of table offset

0000016E 268B6E06                   mov     bp,[es:bp+newsp]      ; get new SP value
                                    ; 21/10/2022
                                    ;mov    bp,[es:bp+6]
```

```
                                ; 11/12/2022
                                ;cmp    [es:bp+0],ax
00000172 26394600               cmp     [es:bp],ax              ; check for offset into table
00000176 7544                   jne     short foundbad02

                                ; 02/07/2023 (MSDOS 6.21 SYSINIT code)
00000178 8CC0                   mov     ax,es                   ; point ss,sp to the new stack
0000017A 8EC5                   mov     es,bp
0000017C 89E5                   mov     bp,sp
0000017E 8B6E06                 mov     bp,[bp+6]
00000181 8ED0                   mov     ss,ax
00000183 8CC4                   mov     sp,es
00000185 8EC0                   mov     es,ax
00000187 2E8B6E00               mov     bp,[cs:bp]

                                ; 21/10/2022 (MSDOS 5.0 SYSINIT code)
                                ;push    bp
                                ;mov     bp,sp
                                ;mov     ax,[bp+8]
                                ;pop     bp
                                ;push    es
                                ;pop     ss
                                ;mov     sp,bp
                                ;mov     bp,ax
                                ; 11/12/2022
                                ;;mov   bp,[cs:bp+0]
                                ;mov    bp,[cs:bp]

0000018B 9C                     pushf                           ; go execute the real interrupt handler
                                ; 11/12/2022
0000018C 2EFF5E00               call    far [cs:bp]             ;  which will iret back to here
                                ; 21/10/2022
                                ;call   far [cs:bp+0]

00000190 89E5                   mov     bp,sp                   ; retrieve the table offset for us
                                ; 11/12/2022
00000192 268B6E00               mov     bp,[es:bp]              ;  but leave it on the stack
                                ; 21/10/2022
                                ;mov    bp,[es:bp+0]
00000196 268E5604                   mov     ss,[es:bp+savedss]      ; get old stack back
0000019A 268B6602                   mov     sp,[es:bp+savedsp]

                                    ; 11/12/2022
                                    ;mov    byte [es:bp+allocbyte],free ; free the entry
                                    ; 21/10/2022
0000019E 26C6460000                 mov     byte [es:bp],free ; 0
000001A3 2E892E[1000]               mov     [cs:nextentry],bp       ; setup to use next time

000001A8 07                         pop     es
000001A9 5D                         pop     bp                      ; saved on entry
000001AA 58                         pop     ax                      ; saved on entry
000001AB 83C402                     add     sp,2
000001AE CF                         iret                            ; done with this interrupt

                                notfree02:
000001AF 3C01                       cmp     al,allocated            ; error flag
000001B1 7404                       je      short findnext02        ;  no, continue
                                    ; 11/12/2022
                                    ;xchg   [es:bp+allocbyte],al    ;  yes, restore error value
                                    ; 21/10/2022
000001B3 26864600                   xchg    [es:bp],al

                                findnext02:
000001B7 E81200                     call    longpath
000001BA EBA8                       jmp     short found02

                                foundbad02:
000001BC 2E3B2E[0C00]               cmp     bp,[cs:firstentry]
000001C1 72F4                       jc      short findnext02
000001C3 89C5                       mov     bp,ax                   ; flag this entry
                                    ; 11/12/2022
                                    ;mov    byte [es:bp+allocbyte],clobbered
                                    ; 21/10/2022
000001C5 26C6460003                 mov     byte [es:bp],clobbered ; 3
000001CA EBEB                       jmp     short findnext02        ; keep looking

                                ; ---------------------------------------------------------------------------

                                ; Common routines

                                longpath:
                                    ; 21/03/2019
000001CC 2E8B2E[0E00]               mov     bp,[cs:lastentry]       ; start with last entry in table
                                lploopp:
                                    ; 11/12/2022
                                    ;cmp    byte [es:bp+allocbyte],free ; is entry free?
                                    ; 21/10/2022
000001D1 26807E0000                 cmp     byte [es:bp],free
000001D6 7512                       jne     short inuse             ;  no, try next one

000001D8 B001                       mov     al,allocated
                                    ; 11/12/2022
                                    ;xchg   [es:bp+allocbyte],al    ; allocate entry
                                    ; 21/10/2022
000001DA 26864600                   xchg    [es:bp],al
000001DE 3C00                       cmp     al,free                 ; is it still free?
000001E0 7414                       je      short found             ;  yes, go use it

000001E2 3C01                       cmp     al,allocated            ; is it other than Allocated or Free?
000001E4 7404                       je      short inuse             ;  no, check the next one

                                    ; 11/12/2022
                                    ;mov    [es:bp+allocbyte],al    ;  yes, put back the error state
                                    ; 21/10/2022
000001E6 26884600                   mov     [es:bp],al
                                inuse:
000001EA 2E3B2E[0C00]               cmp     bp,[cs:firstentry]
000001EF 7406                       je      short fatal
000001F1 83ED08                     sub     bp,entrysize
000001F4 EBDB                       jmp     short lploopp
                                found:
000001F6 C3                         retn
                                fatal:
000001F7 1E                         push    ds
000001F8 B800F0                     mov     ax,0F000h               ;look at the model byte
000001FB 8ED8                       mov     ds,ax
000001FD 803EFEFFF9                 cmp     byte [0FFFEh],0F9h ; mdl_convert ; convertible?
00000202 1F                         pop     ds
00000203 7504                       jne     short skip_nmis

00000205 B007                       mov     al,07h                  ; disable pc convertible nmis
00000207 E672                       out     72h,al

                                skip_nmis:
```

```
00000209 FA                             cli                             ; disable and mask
0000020A B0FF                           mov     al,0FFh                 ;   all other ints
0000020C E621                           out     021h,al
0000020E E6A1                           out     0A1h,al

00000210 8CCE                           mov     si,cs
00000212 8EDE                           mov     ds,si
00000214 BE[3B02]                       mov     si,fatal_msg
                                ;SR;
                                ;   We set all foci to this VM to issue the stack failure message
                                ;
00000217 50                             push    ax
00000218 1E                             push    ds
                                        ;;mov   ax,Bios_Data ; 0070h
                                        ;mov    ax,KERNEL_SEGMENT ; 0070h
                                        ; 21/10/2022
00000219 B87000                         mov     ax,DOSBIODATASEG
0000021C 8ED8                           mov     ds,ax

                                        ;test   byte [08D0h],1        ; (MSDOS 6.21, IO.SYS - SYSINIT:021Eh)
0000021E F606[1208]01                   test    byte [IsWin386],1 ; (retrodos4.sys, offset: ****h)
00000223 1F                             pop     ds
00000224 58                             pop     ax
00000225 7405                           jz      short fatal_loop      ; win386 not present, continue

                                        ;;call  far ptr 0070h:08D1h ; (MSDOS 621, IO.SYS - SYSINIT:0227h)
                                        ;call   KERNEL_SEGMENT:V86_Crit_SetFocus ; set focus to this VM
                                        ; 21/10/2022
00000227 9A[1308]7000                   call    DOSBIODATASEG:V86_Crit_SetFocus ; 0070h:08D1h
                                ;
                                ;SR; We do not bother about the returned status of this call.
                                ;
                                fatal_loop:
0000022C AC                             lodsb
0000022D 3C24                           cmp     al,'$'
0000022F 7408                           je      short fatal_done

00000231 B307                           mov     bl,7
00000233 B40E                           mov     ah,14
00000235 CD10                           int     10h                     ; whoops, this enables ints
00000237 EBF3                           jmp     short fatal_loop

                                fatal_done:
00000239 EBFE                           jmp     short fatal_done


                                ; 21/03/2019 - Retro DOS v4.0 (MSDOS 6.0, SYSINIT1.ASM, 1991)
                                ; ----------------------------------------------------------------------------
                                ;   include msbio.cl5             ; fatal stack error message

                                ; MSDOS 6.21, IO.SYS, SYSINIT:023Bh

                                ; STKMES.INC - MSDOS 3.3 (24/07/1987)
                                ; ----------------------------------------------------------------------------
                                ; 04/06/2018 - Retro DOS v3.0

                                fatal_msg:
0000023B 0D0A                           db      0Dh,0Ah
0000023D 070D0A                         db      7,0Dh,0Ah
00000240 496E7465726E616C20-            db      "Internal stack overflow",0Dh,0Ah
00000249 737461636B206F7665-
00000252 72666C6F770D0A
00000259 53797374656D206861-            db      "System halted",0Dh,0Ah,"$"
00000262 6C7465640D0A24

                                endstackcode:

                                ; ----------------------------------------------------------------------------
                                ; SYINIT1.ASM (MSDOS 6.0, 1991) 'SYSINIT' jump addr from 'MSINIT.ASM'
                                ; ----------------------------------------------------------------------------
                                ; 04/06/2018 - Retro DOS v3.0 (MSDOS 3.3, SYSINIT1.ASM, 24/07/1987)

                                ; 22/03/2019 - Retro DOS v4.0

                                ; SYSINIT:0269h (MSDOS 6.21 IO.SYS, SYSINIT segment, offset: 0269h)

                                ; ('SYSINIT:' location/address is used in 'retrodos4.s'. If following
                                ; address will be changed, it must also be changed in 'retrodos4.s'.)

                                ; 21/10/2022- Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ; ----------------------------------------------------------------------------
                                ; SYSINITSEG:0267h (MSDOS 5.0 IO.SYS, SYSINIT segment, offset: 0267h)

                                ; SYSINIT:0269h (MSDOS 6.22 IO.SYS, SYSINIT segment, offset: 0269h)

                                ; 29/12/2023- Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
                                ; ----------------------------------------------------------------------------
                                ; SYSINITSEG:0269h (PCDOS 7.1 IBMBIO.COM, SYSINIT segment, offset: 0269h)

                                SYSINIT:
00000269 E9AD01                         JMP GOINIT
                                    ;JMP    SYSIN ; 25/02/2018 - Retro DOS 2.0 modification

                                ; ----------------------------------------------------------------------------

                                struc DDHighInfo
00000000 ????????                .ddhigh_CSegPtr resd 1    ; pointer to code segment to be relocated
00000004 ????                    .ddhigh_CSegLen resw 1    ; length of code segment to be relocated
00000006 ????????                .ddhigh_CallBak resd 1    ; pointer to the call back routine
                                endstruc

                                ; 22/03/2019 - Retro DOS v4.0

0000026C 00                     runhigh: db 0

                                ; 02/11/2022
                                ;align 4

                                DOSINFO:
0000026D 00000000                   dd      0       ; address of the DOS Sysini Variables
                                ;MSDOS:
                                dos_temp_location: ; dword ; MSDOS 6.0
                                dosinit:           ; MSDOS 6.0
00000271 0000                       dw      0

                                ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;FINAL_DOS_LOCATION: ; 20/04/2019 - Retro DOS v4.0
                                ;   dw      0
                                ;MSDOS 5.0 IO.SYS - SYSINIT:0271h

                                CURRENT_DOS_LOCATION:
00000273 0000                       dw      0

                                ;DOSSIZE: ; Retro DOS 2.0 feature - 25/02/2018
```

```
23054                                   ;   dw      0   ; 'MSDOS.BIN' kernel size in words
23055
23056                                   ; 22/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
23057                                   ; (MSDOS 5.0 MSDOS.SYS size is 37394 bytes)
23058                                   ;DOSSIZE equ       0A000h  ; (MSDOS 6.0 - SYSINIT1.ASM - 1991)
23059                                   ; 30/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
23060                                   ; 03/09/2023 (PCDOS 7.1 IBMDOS.COM size is 42566 bytes, 04/12/2003)
23061                                   DOSSIZE equ 0B000h  ; (PCDOS 7.1 - SYSINIT)
23062
23063                                   DEVICE_LIST:
23064 00000275 00000000                    dd      0
23065
23066                                   ; 04/06/2018 - Retro DOS v3.0
23067                                   ; 28/03/2018
23068                                   ;; MSDOS 3.3 - SYSINIT1.ASM - 24/07/1987
23069                                   ;
23070                                   sysi_country:
23071 00000279 00000000                    dd      0 ; 5/29/86 Pointer to country table in DOS
23072
23073                                   ; MSDOS 6.0
23074 0000027D 00000000                 dos_segreinit:     dw      0,0     ; room for dword
23075
23076 00000281 0000                     lo_doscod_size:    dw      0       ; dos code size when in low mem
23077 00000283 0000                     hi_doscod_size:    dw      0       ; dos code size when in HMA
23078
23079 00000285 0000                     def_php:    dw      0
23080
23081                                   ; M022--
23082                                   ; pointer for calling into Bios_Code for re-initializing segment values.
23083                                   ;  call with ax = new segment for Bios_Code. Notice that we'll
23084                                   ;  call it in its temporary home, cuz seg_reinit won't get moved to
23085                                   ;  the new home.
23086
23087                                   ;Bios_Code  equ     KERNEL_SEGMENT  ; 0070h
23088                                   ; 21/10/2022
23089                                   ;DOSBIOCODESEG     equ     02C7h ; (MSDOS 5.0 IO.SYS)
23090
23091                                   ; 22/10/2022
23092                                   seg_reinit_ptr:    ; label dword
23093 00000287 [3200]                             dw      seg_reinit ; Bios_Code:0032h for MSDOS 6.21 IO.SYS
23094                                   temp_bcode_seg:
23095                                             ;dw     Bios_Code  ; 02CCh for MSDOS 6.21 IO.SYS
23096                                             ; 22/10/2022
23097 00000289 2D03                               dw      DOSBIOCODESEG ; 02C7h for MSDOS 5.0 IO.SYS
23098                                                     ; 364h for PCDOS 7.1 IBMBIO.COM - 29/12/2023
23099                                   fake_floppy_drv:
23100 0000028B 00                                 db      0       ; set to 1 if this machine
23101                                                     ; does not have any floppies!!!
23102
23103                                   ; Internal Stack Parameters
23104
23105 0000028C 0900                     stack_count:       dW      defaultcount ; 9
23106 0000028E 8000                     stack_size: dw     defaultsize  ; 128
23107 00000290 00000000                 stack_addr: dd     0
23108
23109                                   ; 05/06/2018 - Retro DOS v3.0
23110
23111                                   ; various default values
23112
23113 00000294 0100                     MEMORY_SIZE:       dw      1
23114
23115                                   ; 22/03/2019 - Retro DOS v4.0 (MSDOS 6.0 source, MSDOS 6.21 disassembled src.)
23116
23117 00000296 0000                     RPLMemTop:  dw      0  ; 22/10/2022 (MSDOS 5.0 IO.SYS SYSINIT:0294h)
23118 00000298 00                       DEFAULT_DRIVE:     db      0       ; initialized by ibminit.
23119 00000299 FFFF                     buffers:    dw      0FFFFh  ; initialized during buffer allocation
23120 0000029B 0000                     h_buffers:  dw      0       ; # of the heuristic buffers. initially 0.
23121 0000029D 0000                     singlebuffersize: dw      0       ; maximum sector size + buffer head
23122
23123 0000029F 08                       FILES:      db      8       ; enough files for pipe
23124 000002A0 04                       FCBS:       db      4       ; performance for recycling
23125 000002A1 00                       KEEP:       db      0       ; keep original set
23126 000002A2 05                       NUM_CDS: db 5       ; 5 net drives
23127
23128                                   ; 22/10/2022 (MSDOS 5.0 SYSINIT)
23129                                   ;;CONFBOT: dw      0
23130                                   ;;ALLOCLIM: dw     0
23131                                   ;CONFBOT: ; 02/11/2022
23132                                   ;top_of_cdss: dw 0
23133
23134                                   ; 30/12/2022 - Retrodos v4.2 (MSDOS 6.21 SYSINIT)
23135                                   ; (SYSINIT:02A3h)
23136 000002A3 0000                     CONFBOT: dw 0
23137 000002A5 0000                     ALLOCLIM: dw       0
23138 000002A7 0000                     top_of_cdss: dw 0
23139
23140                                   ; 02/11/2022 (MSDOS 5.0 SYSINIT)
23141                                   ; 30/12/2022 (MSDOS 6.21 SYSINIT)
23142                                   ;ALLOCLIM: dw      0       ; (SYSINIT:02A3h)
23143
23144 000002A9 413A5C00                 DirStrng: db       "A:\",0 ; string for the root directory of a drive
23145
23146                                   ; 30/12/2022 - Retro DOS v4.2 (MSDOS 6.21 SYSINIT)
23147                                   %if 0
23148                                   ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
23149                                   ; (SYSINIT:02A9h)
23150
23151                                   command_line:
23152                                       db      2,0
23153                                       db      'P'
23154                                       db      0
23155                                       times   124 db 0 ; db 124 dup(0)
23156
23157                                   %endif
23158
23159                                   ; (SYSINIT:0329h)
23160 000002AD 00                       ZERO:       db      0
23161 000002AE 00                       sepchr:     db      0
23162 000002AF 0000                     linecount: dw      0                       ; line count in config.sys
23163 000002B1 20202020200D0A24         showcount: db      '     ',cr,lf,'$'       ; used to convert linecount to ascii.
23164 000002B9 0000                     buffer_linenum: dw 0               ; line count for "buffers=" command if entered.
23165
23166 000002BB FF                       sys_model_byte:    db      0FFh            ; model byte used in sysinit
23167 000002BC 00                       sys_scnd_model_byte: db 0          ; secondary model byte used in sysinit
23168
23169 000002BD 0000                     buf_prev_off:      dw      0
23170
23171                                           ;IF      NOT NOEXEC
23172                                   ;COMEXE EXEC0 <0,COMMAND_LINE,DEFAULT_DRIVE,ZERO>
23173                                           ;ENDIF
23174
23175                                   ; 29/12/2023
23176                                   ; 01/05/2018
23177                                   COMEXE:
```

```
23178 000002BF 0000                   EXEC0.ENVIRON:     dw      0       ; seg addr of environment
23179 000002C1 [BB4B]                 EXEC0.COM_LINE:    dw      command_line ; pointer to asciz command line
23180 000002C3 0000                               dw      0       ; SYSINIT segment (0473h for MSDOS 6.21 IO.SYS)
23181                                                             ; SYSINIT segment (0544h for PCDOS 7.1 IBMBIO.COM)
23182 000002C5 [9802]                 EXEC0.5C_FCB:      dw      DEFAULT_DRIVE ; default fcb at 5C
23183 000002C7 0000                               dw      0       ; SYSINIT segment (0473h for MSDOS 6.21 IO.SYS)
23184 000002C9 [AD02]                 EXEC0.6C_FCB:      dw      ZERO    ; default fcb at 6C
23185 000002CB 0000                               dw      0
23186
23187                                 ; variables for install= command.
23188
23189 000002CD 00                     multi_pass_id:     db      0               ; parameter passed to multi_pass
23190                                                                   ;  indicating the pass number
23191                                                                   ; 0 - do scan for DOS=HIGH/LOW
23192                                                                   ; 1 - load device drivers
23193                                                                   ; 2 - was to load IFS
23194                                                                   ;       now it is unused
23195                                                                   ; 3 - do install=
23196                                                                   ; >3 - nop
23197 000002CE 0000                   install_flag:      dw      0
23198
23199                                 have_install_cmd equ      00000001b       ; config.sys has install= commands
23200                                 has_installed      equ    00000010b       ; sysinit_base installed.
23201
23202 000002D0 0000                   config_size:       dw      0               ; size of config.sys file. set by sysconf.asm
23203 000002D2 00000000               sysinit_base_ptr: dd       0               ; pointer to sysinit_base
23204 000002D6 00000000               sysinit_ptr:       dd      0               ; returning addr. from sysinit_base
23205 000002DA 0000                   checksum:   dw     0               ; used by sum_up
23206
23207 000002DC 20<rep 14h>            ldexec_fcb: times 20 db 20h ; db 20 dup (' ') ;big enough
23208 000002F0 00                     ldexec_line:       db      0               ;# of parm characters
23209 000002F1 20                     ldexec_start:      db      ' '
23210 000002F2 00<rep 50h>            ldexec_parm:       times 80 db 0  ; db 80 dup (0)
23211
23212                                 ;instexe exec0     <0,ldexec_line,ldexec_fcb,ldexec_fcb>
23213
23214                                 instexe:
23215 00000342 0000                   iexec.environ:     dw      0               ; seg addr of environment
23216 00000344 [F002]                 iexec.ldexec_line: dw      ldexec_line ; pointer to asciz command line
23217 00000346 0000                               dw      0       ; SYSINIT segment (0473h for MSDOS 6.21 IO.SYS)
23218                                                             ; SYSINIT segment (0544h for PCDOS 7.1 IBMBIO.COM)
23219 00000348 [DC02]                 iexec.ldexec_5c_fcb: dw    ldexec_fcb      ; default fcb at 5C
23220 0000034A 0000                               dw      0       ; SYSINIT segment (0473h for MSDOS 6.22 IO.SYS)
23221 0000034C [DC02]                 iexec.ldexec_6c_fcb: dw    ldexec_fcb      ; default fcb at 6C
23222 0000034E 0000                               dw      0
23223
23224                                 ; variables for comment=
23225
23226 00000350 00                     com_level:  db     0               ; level of " " in command line
23227 00000351 00                     cmmt:              db      0               ; length of comment string token
23228 00000352 00                     cmmt1:             db      0               ; token
23229 00000353 00                     cmmt2:             db      0               ; token
23230 00000354 00                     cmd_indicator:     db      0
23231 00000355 00                     donotshownum:      db      0
23232
23233 00000356 0000                   count:             dw      0
23234 00000358 0000                   org_count:  dw     0
23235 0000035A 0000                   chrptr:            dw      0
23236 0000035C 0000                   cntryfilehandle: dw        0
23237 0000035E 0000                   old_area:   dw     0
23238 00000360 0000                   impossible_owner_size: dw 0               ; paragraph
23239
23240                                 bucketptr: ; label dword
23241                                 bufptr:        ; label dword              ; leave this stuff in order!
23242 00000362 0000                   memlo:     dw      0
23243                                 prmblk:        ; label word
23244 00000364 0000                   memhi:     dw      0
23245 00000366 0000                   ldoff:     dw      0
23246 00000368 0000                   area:      dw      0
23247
23248                                 ; 29/12/2023 - PCDOS 7.1 IBMBIO.COM - SYSINIT:036Ah
23249 0000036A 0000                   prev_memhi: dw 0
23250 0000036C 0000                   prev_alloclim:     dw 0
23251 0000036E 00                     dosdata_umb:       db 0
23252
23253                                 ; Following is the request packet used to call INIT routines for
23254                                 ; all device drivers. Some fields may be accessed individually in
23255                                 ; the code, and hence have individual labels, but they should not
23256                                 ; be separated.
23257
23258 0000036F 19                     packet:    db      25                      ; PCDOS 7.1 IBMBIO.COM
23259                                     ;db    24                      ; was 22
23260 00000370 00                       db     0
23261 00000371 00                       db     0                       ; initialize code
23262 00000372 0000                     dw     0
23263 00000374 00<rep 8h>               times  8 db 0  ; db 8 dup (?)
23264
23265 0000037C 00                     unitcount:  db     0
23266 0000037D 00000000               break_addr: dd     0
23267 00000381 00000000               bpb_addr:   dd     0
23268                                 drivenumber:       ; 22/10/2022
23269 00000385 00                     devdrivenum:       db      0
23270 00000386 0000                   configmsgflag:     dw      0  ; used to control "error in config.sys line #" message
23271
23272                                 ; end of request packet
23273
23274                                 ;drivenumber:      db      0  ; 22/03/2019
23275
23276                                 toomanydrivesflag:
23277 00000388 00                               db      0  ; >24 fixed disk partitions flag ; M029
23278 00000389 90                     align 2
23279
23280                                 BCodeSeg:   ; 21/10/2022
23281 0000038A 2D03                     dw      DOSBIOCODESEG ; (02C7h for MSDOS 5.0 IO.SYS)
23282                                                   ; 0364h for PCDOS 7.1 IBMBIO.COM - 29/12/2023
23283                                   ;dw     Bios_Code  ; = KERNEL_SEGMENT = 0070h (for Retro DOS v4.0)
23284                                                     ; BCodeSeg = 2CCh (for MSDOS 6.21 IO.SYS)
23285
23286                                 ; 30/12/2022
23287                                 ; MSDOS 6.21 IO.SYS, SYSINIT:0387h
23288                                 ;
23289                                 ; Magicbackdoor: dd 0
23290                                 ; NullBackdoor:
23291                                 ;           retf
23292
23293                                 ; 02/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
23294                                 ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
23295                                 ; 19/04/2019
23296                                 _timer_lw_:
23297 0000038C 0000                     dw      0  ; MSDOS 6.21 IO.SYS - SYSINIT:038Ch
23298
23299                                 ; 29/12/2023 - Retro DOS v5.0
23300                                 ; PCDOS 7.1 IBMBIO.COM - SYSINIT:038Eh
23301
```

```
23302 0000038E 00                       F5_key:     db 0
23303 0000038F 00                       F8_key:     db 0
23304 00000390 00000000                 MagicBackdoor:     dd 0
23305                                   NullBackdoor:
23306 00000394 CB                                   retf
23307
23308                                   ;SR;
23309                                   ; This is the communication block between the DOS and the BIOS. It starts at
23310                                   ;the SysinitPresent flag. Any other data that needs to be communicated
23311                                   ;to the DOS should be added after SysinitPresent. The pointer to this block
23312                                   ;is passed to DOS as part of the DOSINIT call.
23313                                   ;
23314
23315                                   BiosComBlock:
23316                                       ;dd     Bios_Data:SysinitPresent
23317                                               ; 0070h:08FDh for MSDOS 6.21 IO.SYS
23318 00000395 [DD07]                     dw      SysinitPresent  ; (retrodos4.sys, offset: ****h)
23319                                       ;dw     KERNEL_SEGMENT ; 0070h
23320                                       ; 21/10/2022
23321 00000397 7000                       dw      DOSBIODATASEG ; 0070h
23322
23323                                   ;align 2
23324
23325                                       ; 22/10/2022 - (MSDOS 5.0 IO.SYS, SYSINIT:0406h)
23326                                       ; 30/12/2022 - (MSDOS 6.21 IO.SYS, SYSINIT:0392h)
23327                                   tempstack:
23328 00000399 00<rep 80h>                ;    times   128 db 0  ; db 80h dup (?)
23329
23330                                   ; ------------------------------------------------------------------------------
23331
23332                                       ; 29/12/2023 - Retro DOS v5.0
23333                                       ; 22/10/2022 - Retro DOS v4.0
23334                                       ;       ; (MSDOS 5.0 IO.SYS, SYSINIT:0486h)
23335                                   GOINIT:             ; (MSDOS 6.22 IO.SYS, SYSINIT:0412h)
23336                                               ; (PCDOS 7.1 IBMBIO.COM, SYSINIT:0419h)
23337                                       ; 12/12/2023
23338 00000419 0E                         push    cs
23339 0000041A 1F                         pop     ds
23340
23341                                       ; 12/12/2022
23342                                       ; 22/03/2019 - Retro DOS v4.0
23343                                       ; 06/07/2018
23344                                       ; 04/06/2018 - Retro DOS v3.0
23345                                   ; before doing anything else, let's set the model byte
23346 0000041B B4C0                       mov     ah,0C0h                 ; get system configuration
23347 0000041D CD15                       int     15h                     ;
23348 0000041F 7214                       jc      short no_rom_config
23349
23350                                     ;cmp     ah,0                    ; double check
23351                                     ;jne     short no_rom_config
23352                                     ; 03/09/2023
23353 00000421 08E4                       or      ah,ah
23354 00000423 7510                       jnz     short no_rom_config
23355
23356                                     ; 12/12/2023 ; *
23357                                     ; ds = cs
23358
23359 00000425 268A4702                   mov     al,[es:bx+ROMBIOS_DESC.bios_sd_modelbyte]
23360                                     ;mov     [cs:sys_model_byte],al
23361 00000429 A2[BB02]                   mov     [sys_model_byte],al ; *
23362 0000042C 268A4703                   mov     al,[es:bx+ROMBIOS_DESC.bios_sd_scnd_modelbyte]
23363                                     ;mov     [cs:sys_scnd_model_byte],al
23364 00000430 A2[BC02]                   mov     [sys_scnd_model_byte],al ; *
23365                                     ;jmp     short SYSIN
23366                                     ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
23367 00000433 EB29                       jmp     short move_myself
23368
23369                                   no_rom_config:                          ; Old ROM
23370                                       ; 12/12/2023
23371                                     ;mov     ax,0F000h
23372                                     ;mov     ds,ax
23373                                     ;mov     al,[0FFFEh]
23374                                     ;mov     [cs:sys_model_byte],al; set the model byte.
23375                                     ; 12/12/2023
23376                                     ; ds = cs
23377 00000435 B800F0                     mov     ax,0F000h
23378 00000438 8EC0                       mov     es,ax
23379 0000043A 26A0FEFF                   mov     al,[es:0FFFEh]
23380 0000043E A2[BB02]                   mov     [sys_model_byte],al     ; set the model byte.
23381
23382                                   ; set fake_floppy_drv if there is no diskette drives in this machine.
23383                                   ; execute the equipment determination interrupt and then
23384                                   ; check the returned value to see if we have any floppy drives
23385                                   ; if we have no floppy drive we set cs:fake_floppy_drv to 1
23386                                   ; see the at tech ref bios listings for help on the equipment
23387                                   ; flag interrupt (11h)
23388
23389                                       ; 22/10/2022
23390                                   ;check_for_fake_floppy:                 ; entry point for rom_config above
23391 00000441 CD11                       int     11h                     ; check equipment flag
23392
23393                                       ; 29/12/2023 - Retro DOS v5.0
23394                                       ;jmp    short check_for_fake_floppy
23395                                       ; PCDOS 7.1 IBMBIO.COM - SYSINIT:0446h
23396                                       ;db     52h     ; 'RPS' sign
23397                                       ;db     50h
23398                                       ;db     53h
23399
23400                                   check_for_fake_floppy:
23401                                       ; 29/12/2023
23402                                       ; PCDOS 7.1 IBMBIO.COM - SYSINIT:0449h
23403                                       ;or      ax, 1          ; (nonsense! this may be overwritten/disabled
23404                                       ;                       ; by using 'RPS' sign position)
23405                                       ;                       ;               03/07/2023 - Erdogan Tan
23406                                       ;test    ax, 1          ; have any floppies?
23407
23408                                       ; 12/12/2022
23409 00000443 A801                       test    al,1
23410                                     ;test    ax,1                    ; have any floppies?
23411 00000445 7517                       jnz     short move_myself       ; yes,normal system
23412
23413                                   ; Some ROM BIOSs lie that there are no floppy drives. Lets find out
23414                                   ; whether it is an old ROM BIOS or a new one
23415                                   ;
23416                                   ; WARNING !!!
23417                                   ;
23418                                   ; This sequence of code is present in MSINIT.ASM also. Any modification
23419                                   ; here will require an equivalent modification in MSINIT.ASM also
23420
23421                                       ; 12/12/2023
23422                                       ;push    es  ; not necessary
23423
23424 00000447 30C9                       xor     cl,cl
23425 00000449 B408                       mov     ah,8                    ; get disk parameters
```

```
                                    mov     dl,0                    ; of drive 0
                                    int     13h

                                    ;pop    es  ; 12/12/2023

                                    jc      short move_myself       ; if error lets assume that the
                                                                    ;  ROM BIOS lied
                                    ;cmp    cl,0                    ; double check (max sec no cannot be 0)
                                    ;je     short move_myself
                                    ; 03/09/2023
                                    or      cl,cl
                                    jz      short move_myself

                                    or      dl,dl                   ; number of flp drvs == 0?
                                    jnz     short move_myself       ; no

                                    ;mov    byte [cs:fake_floppy_drv],1 ; set fake flag.
                                    ; 12/12/2023
                                    ; ds = cs
                                    mov     byte [fake_floppy_drv],1 ; set fake flag.

                                move_myself:
                                    ; 12/12/2023
                                    ;cld    ; not necessary             ; set up move
                                    ;xor    si,si
                                    ;mov    di,si

                                    ; 12/12/2023
                                    ; ds = cs
                                    ; 12/12/2022
                                    ;push   cs
                                    ;pop    ds

                                    ;mov    cx,[cs:MEMORY_SIZE]
                                    mov     cx,[MEMORY_SIZE] ; 12/12/2022

                                    ; (MSDOS 6.0 - SYSINIT1.ASM - 1991)
                                ;;; if      msver
                                ;   cmp     cx,1            ; 1 means do scan
                                ;   jnz     short noscan
                                ;   mov     cx,2048         ; start scanning at 32k boundary
                                ;   xor     bx,bx
                                ;
                                ;memscan:inc       cx
                                ;   jz      short setend
                                ;   mov     ds,cx
                                ;   mov     al,[bx]
                                ;   not     al
                                ;   mov     [bx],al
                                ;   cmp     al,[bx]
                                ;   not     al
                                ;   mov     [bx],al
                                ;   jz      short memscan
                                ;setend:
                                ;   mov     cs:[memory_size],cx
                                ;;; endif

                                ;noscan:                        ; cx is mem size in para
                                ;;
                                ;;  cas -- a) if we got our memory size from the ROM, we should test it
                                ;;           before we try to run.
                                ;;         b) in any case, we should check for sufficient memory and give
                                ;;           an appropriate error diagnostic if there isn't enough
                                ;
                                ;   push    cs
                                ;   pop     ds
                                ;
                                ;;  cas note:  It would be better to put dos + bios_code BELOW sysinit
                                ;;    that way it would be easier to slide them down home in a minimal
                                ;;    memory system after sysinit.  As it is, you need room to keep
                                ;;    two full non-overlapping copies, since sysinit sits between the
                                ;;    temporary home and the final one.  the problem with doing that
                                ;;    is that sys*.asm are filled with "mov ax,cs, sub ax,11h" type stuff.
                                ;
                                ;   dec     cx                  ; one para for an arena at end of mem
                                ;                               ; in case of UMBs

                                    ; 22/10/2022
                                    ; (MSDOS 5.0 IO.SYS SYSINIT:04DBh)

                                    ; 12/12/2022
                                    ;push   cs
                                    ;pop    ds

                                    dec     cx

                                ;------ Check if an RPL program is present at TOM and do not tromp over it

                                    xor     bx,bx
                                    mov     es,bx
                                    ;mov    bx,[es:(2Fh*4)] ; INT 2Fh address (0:0BCh)
                                    ;mov    es,[es:((2Fh*4)+2)] ; INT 2Fh segment (0:0BEh)
                                    ; 29/09/2023
                                    les     bx,[es:(2Fh*4)]
                                    cmp     word [es:bx+3],'RP'
                                    jne     short NoRPL
                                    cmp     byte [es:bx+5],'L'
                                    jne     short NoRPL

                                    mov     dx,cx                   ; get TOM into DX
                                    push    dx
                                    mov     ax,4A06h
                                    ;mov    ax,(multMULT<<8)+multMULTRPLTOM
                                    int     2Fh                     ; Get new TOM from any RPL
                                    pop     ax
                                    mov     cx,dx
                                    cmp     dx,ax
                                    je      short NoRPL

                                    ; 11/12/2022
                                    ; ds = cs
                                    mov     [RPLMemTop],dx
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;mov    [cs:RPLMemTop],dx

                                    dec     cx
                                NoRPL:
                                    mov     ax,SI_end               ; need this much room for sysinit
                                                                    ; (SI_end == sysinit code size)
                                                                    ; 03/09/2023
                                                                    ; (58A0h for MSDOS 6.21 IO.SYS)
                                                                    ; (5B40h for PCDOS 7.1 IBMBIO.COM)
                                    call    off_to_para
                                    sub     cx,ax
```

```
23550
23551                                   ; we need to leave room for the DOS and (if not ROMDOS) for the BIOS
23552                                   ; code above sysinit in memory
23553                                   ;
23554 00000497 81E9000B                     sub     cx,DOSSIZE/16 ; (0A00h)       ; leave this much room for DOS
23555                                                               ; (0B00h) ; (PCDOS 7.1 IBMBIO.COM) -03/09/2023-
23556
23557 0000049B B8701D                       mov     ax,BCODE_END            ; (1A60h for MSDOS 5.0 IO.SYS)
23558                                                                       ; (1A70h for MSDOS 6.21 IO.SYS)
23559                                                                       ; 03/09/2023
23560                                                                       ; (1E00h for PCDOS 7.1 IBMBIO.COM)
23561 0000049E E8F908                       call    off_to_para             ; leave this much room for BIOS code
23562 000004A1 29C1                         sub     cx,ax
23563 000004A3 8EC1                         mov     es,cx                   ; segment where sysinit will be located
23564
23565                                       ; 12/12/2023
23566 000004A5 FC                           cld     ; not necessary               ; set up move
23567 000004A6 31F6                         xor     si,si
23568 000004A8 89F7                         mov     di,si
23569
23570 000004AA B9[1054]                     mov     cx,SI_end               ; (sysinit code size)
23571 000004AD D1E9                         shr     cx,1                    ; divide by 2 to get words
23572 000004AF F3A5                         rep     movsw                   ; relocate sysinit
23573
23574 000004B1 06                           push    es                      ; push relocated segment
23575 000004B2 B8[B704]                     mov     ax,SYSIN
23576 000004B5 50                           push    ax                      ; push relocated entry point
23577
23578 000004B6 CB                           retf                            ; far jump to relocated sysinit
23579
23580                                   ; =============== S U B R O U T I N E =======================================
23581
23582                                   ; 30/12/2023
23583                                   ; PCDOS 7.1 IBMBIO.COM - SYSINIT:04CEh
23584                                   %if 0
23585                                   get_cpu_type:
23586                                       pushf
23587                                       push    bx
23588                                       xor     bx,bx
23589                                       xor     ax,ax
23590                                       push    ax
23591                                       popf
23592                                       pushf
23593                                       pop     ax
23594                                       and     ax,0F000h
23595                                       cmp     ax,0F000h
23596                                       jz      short cpu_8086
23597                                       mov     ax,0F000h
23598                                       push    ax
23599                                       popf
23600                                       pushf
23601                                       pop     ax
23602                                       and     ax,0F000h
23603                                       jz      short cpu_286
23604                                   cpu_386:
23605                                       inc     bx
23606                                   cpu_286:
23607                                       inc     bx
23608                                   cpu_8086:
23609                                       mov     ax,bx
23610                                       pop     bx
23611                                       popf
23612                                       retn
23613                                   %endif
23614
23615                                   ; ---------------------------------------------------------------------------
23616
23617                                   ;   MOVE THE DOS TO ITS PROPER LOCATION
23618
23619                                       ; 22/10/2022 - Retro DOS 4.0 (Modified MSDOS 5.0 IO.SYS)
23620                                       ; (SYSINIT:0533h)
23621                                       ; 30/12/2022 - Retro DOS 4.2 (Modified MSDOS 6.21 IO.SYS)
23622                                       ; (SYSINIT:04BFh)
23623                                       ; 03/09/2023 - Retro DOS 4.2 (5.0 - Modified PCDOS 7.1 IBMBIO.COM)
23624                                       ; (SYSINIT:04F3h)
23625                                   SYSIN:
23626                                       ; Retro DOS 5.0 - 30/12/2023
23627                                       ; Retro DOS 4.0 - 22/03/2019
23628                                       ; Retro DOS 2.0 - 25/02/2018
23629
23630                                       ; 23/04/2019
23631                                       ;;mov   ax,Bios_Data
23632                                       ;mov    ax,KERNEL_SEGMENT ; 0070h
23633                                       ; 21/10/2022
23634 000004B7 B87000                       mov     ax,DOSBIODATASEG ; 0070h
23635 000004BA 8ED8                         mov     ds,ax
23636
23637                                       ; 30/12/2023 - Retro DOS v5.0
23638                                       ;;;
23639                                       ;push   es
23640                                       ;push   ax              ; not needed (*) E.TAN - 03/07/2023
23641                                       ;push   di
23642
23643                                       ;call   get_cpu_type    ; determine if 386 system
23644                                       ;
23645                                   get_cpu_type:
23646 000004BC 9C                           pushf
23647 000004BD 31C0                         xor     ax,ax
23648 000004BF 50                           push    ax
23649 000004C0 9D                           popf
23650 000004C1 9C                           pushf
23651 000004C2 58                           pop     ax
23652 000004C3 2500F0                       and     ax,0F000h
23653 000004C6 3D00F0                       cmp     ax,0F000h
23654 000004C9 740F                         jz      short cpu_8086
23655 000004CB B800F0                       mov     ax,0F000h
23656 000004CE 50                           push    ax
23657 000004CF 9D                           popf
23658 000004D0 9C                           pushf
23659 000004D1 58                           pop     ax
23660 000004D2 2500F0                       and     ax,0F000h
23661 000004D5 7402                         jz      short cpu_286
23662                                   cpu_386:
23663 000004D7 29C0                         sub     ax,ax
23664                                   cpu_286:
23665 000004D9 40                           inc     ax
23666                                   cpu_8086:   ; ax = 0
23667                                       ; 30/12/2023 - Retro DOS v5.0
23668 000004DA 2EA2[B606]                   mov     [cs:cpu_type],al ; 07/04/2024
23669 000004DE 9D                           popf
23670                                       ;
23671                                       ;cmp    ax,2            ; 0 = 8086, 1 = 286, 2 = 386
23672 000004DF 3C02                         cmp     al,2
23673 000004E1 7512                         jnz     short not_386_system
```

```
000004E3 FC                         cld                     ; 80386
000004E4 1E                         push    ds
000004E5 07                         pop     es              ; change A20 line on/off check code
000004E6 BF[4D07]                   mov     di,cpu386_cmpsd
000004E9 B8B904                     mov     ax,04B9h        ; mov cx,4 ; B90400
000004EC AB                         stosw
000004ED B800F3                     mov     ax,0F300h       ; repz  ; F3
000004F0 AB                         stosw
000004F1 B866A7                     mov     ax,0A766h       ; cmpsd ; 66A7
000004F4 AB                         stosw
                                not_386_system:
                                    ;pop    di
                                    ;pop    ax
                                    ;pop    es
                                    ;;;

000004F5 8C0E[DB07]                 mov     [MoveDOSIntoHMA+2],cs ; set seg of routine to move DOS
000004F9 C606[DD07]01               mov     byte [SysinitPresent],1       ; flag that MoveDOSIntoHMA can be called

                                ; first move the MSDOS.SYS image up to a harmless place
                                ; on top of our new sysinitseg

                                    ; 22/10/2022
000004FE B8[1054]                   mov     ax,SI_end               ; how big is sysinitseg?
00000501 E89608                     call    off_to_para
00000504 8CC9                       mov     cx,cs                   ; pick a buffer for msdos above us
00000506 01C8                       add     ax,cx
00000508 8EC0                       mov     es,ax

0000050A 31F6                       xor     si,si
0000050C 89F7                       mov     di,si

0000050E 2E8E1E[7302]               mov     ds,[cs:CURRENT_DOS_LOCATION] ; where it is (set by msinit)

                                    ;mov    ax,cs
                                    ;mov    ds,ax

                                    ;;;mov  cx,20480  ; MSDOS 6.21 IO.SYS - SYSINIT:04E2h
                                    ;;mov   cx,dossize/2 ; MSDOS 6.0
                                    ;mov    cx,[DOSSIZE] ; words (not bytes!)  ; Retro DOS v4.0 (3.0, 2.0)
                                    ;mov    es,[FINAL_DOS_LOCATION] ; on top of SYSINIT code
                                    ;mov    ds,[CURRENT_DOS_LOCATION]

                                    ; 22/10/2022
00000513 B90058                     mov     cx,DOSSIZE/2 ; 5000h
                                                         ; 03/09/2023
                                                         ; 5800h (PCDOS 7.1)
00000516 F3A5                       rep      movsw
00000518 2E8C06[7302]               mov     [cs:CURRENT_DOS_LOCATION],es

                                ; The DOS code is ORGed at a non-zero value to allow it to be located in
                                ; HIMEM. Thus, the DOS segment location must be adjusted accordingly.
                                ; If this is ROMDOS, however, only the init code is loaded into RAM, so
                                ; this ORG is not done. The entry point is at offset zero in the segment.

                                    ; 22/04/2019 (MSDOS 6.0 & MSDOS 6.21 kernel address modification)
                                    ;mov    ax,cs
                                    ; 02/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS, SYSINIT)
                                    ;mov    ds,ax

                                ; 29/04/2019 - Retro DOS v4.0 ! important MODIFICATION !

                                ;   ; 24/04/2019
                                ;;ifndef ROMDOS
                                ;   mov     ax,[es:3]               ; get offset of dos
                                ;           ; ax = 3DE0h for MSDOS 6.21 kernel (MSDOS.SYS, offset 3)
                                ;   mov     [dosinit],ax            ; that's the entry point offset
                                ;   call    off_to_para             ; subtract this much from segment
                                ;   ; 23/04/2019
                                ;   ;sub    [CURRENT_DOS_LOCATION],ax
                                ;   sub     [FINAL_DOS_LOCATION],ax
                                ;;else
                                ;;  mov     word [dosinit],0        ; entry to init is at zero
                                ;;
                                ;;endif ; ROMDOS

                                    ; 29/04/2019 - Retro DOS v4.0 ! important MODIFICATION !
                                    ; (! MSDOS6.BIN starts with DOSDATA ! - Retro DOS v4.0 modification)

                                    ;mov    ax,[es:0] ; DOSCODE start address = DOSDATA size (= 136Ah)
                                    ;                 ; (Valid for Retro DOS v4.0 only!)

                                    ; 22/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS, SYSINIT)
                                    ; (SYSINIT:0563h for MSDOS 5.0 IO.SYS SYSINIT)
                                    ; 03/09/2023 - Retro DOS 4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:04ECh for MSDOS 6.21 IO.SYS SYSINIT)
                                    ; (SYSINIT:0540h for PCDOS 7.1 IBMBIO.COM SYSINIT)
0000051D A10300                     mov     ax,[3]          ; mov ax, word ptr ds:3
                                                            ; 30/12/2023
                                                            ; ax = 3F10h for IBMDOS 7.1 kernel
                                                            ;  (IBMDOS.SYS, offset 3)

00000520 2EA3[7102]                 mov     [cs:dosinit],ax ; (SYSINIT:0563h for MSDOS 5.0 IO.SYS SYSINIT)
                                    ; 02/11/2022
00000524 E87308                     call    off_to_para             ; subtract this much from segment
00000527 2E2906[7302]               sub     [cs:CURRENT_DOS_LOCATION],ax

                                    ; Current DOSCODE start address = dword [dosinit]

                                ;; If this is not ROMDOS, then the BIOS code is moved to the top of memory
                                ;; until it is determined whether it will be running in HIMEM or not.

                                ;ifndef ROMDOS

                                ; now put Bios_Code up on top of that. Assume Bios_Code + dossize < 64k

                                    ; 22/10/2022
0000052C 8CC0                       mov     ax,es
0000052E 05000B                     add     ax,DOSSIZE/16           ; get paragraph of end of dos
00000531 8EC0                       mov     es,ax
00000533 2E8706[8902]               xchg    ax,[cs:temp_bcode_seg]; swap with original home of Bios_Code
00000538 8ED8                       mov     ds,ax                   ; point to loaded image of Bios_Code

                                    ;mov    si,BCODE_START ; mov si,30h
                                    ; 09/12/2022
0000053A BE[3000]                   mov     si,BCODESTART
                                    ; 02/11/2022
0000053D 89F7                       mov     di,si
                                    ; 30/12/2023
                                    ;mov    cx,1E00h        ; BCODE_END = (SYSINITSEG-DOSBIOCODESEG)*16
                                    ;                       ; (544h-364h)*10h = 1E00h (for PCDOS 7.1 IBMBIO.COM)
                                    ;mov    cx,BCODE_END    ; mov cx,1A60h ; mov cx,1A70h ; 30/12/2022
                                    ;sub    cx,si
                                    ; 31/03/2024
```

```
                                    BCODESIZE equ BCODEEND-BCODESTART
0000053F B9401D                     mov     cx,BCODESIZE
00000542 D1E9                       shr     cx,1
00000544 F3A5                       rep     movsw                   ; move Bios_Code into place

00000546 8CC0                       mov     ax,es                   ; tell it what segment it's in
00000548 2EFF1E[8702]               call    far [cs:seg_reinit_ptr]       ; far call to seg_reinit in Bios_Code (M022)

                                ;endif      ; not ROMDOS

                                ; now call dosinit while it's in its temporary home

                                    ;mov    ax,cs
                                    ;mov    ds,ax

                                    ;mov    dx,[MEMORY_SIZE]        ; set for call to dosinit

                                    ; 22/10/2022

0000054D 2EC43E[9503]               les     di,[cs:BiosComBlock]  ; ptr to BIOS communication block
                                            ; es = KERNEL_SEGMENT (70h), di = 'SysInitPresent' address
00000552 2EC536[7502]               lds     si,[cs:DEVICE_LIST]   ; set for call to dosinit
                                            ; ds = KERNEL_SEGMENT (70h), si = 'res_dev_list' address

00000557 2E8B16[9402]               mov     dx,[cs:MEMORY_SIZE]   ; set for call to dosinit

0000055C FA                         cli
0000055D 8CC8                       mov     ax,cs
0000055F 8ED0                       mov     ss,ax

                                ; 30/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                ; 22/03/2019 - Retro DOS v4.0 (MSDOS 6.0, SYSINIT1.ASM)
                                %define locstack ($ - SYSINIT$) & 0FFFEh  ; 532h in MSDOS 6.21 IO.SYS
                                                                        ; 5A6h in MSDOS 5.0 IO.SYS SYSINIT
                                                                        ; 586h in PCDOS 7.1 IBMBIO.COM SYSINIT
                                ;SYSINIT:0532h:

                                ; 22/10/2022
                                ; ------------------------------------------------------------------------------
                                ;SYSINIT:05A6h:
                                ;locstack:  ; (at SYSINIT:05A6h for MSDOS 5.0 IO.SYS)

                                ; 03/09/2023
                                ; (locstack at SYSINIT:0586h in PCDOS 7.1 IBMBIO.COM SYSINIT)

                                    ;mov    sp,05A6h
00000561 BC6005                     mov     sp,locstack           ; set stack

00000564 FB                         sti

                                ;align 2
                                    ; 30/03/2018
                                ;LOCSTACK:
                                       ;CALL      FAR [CS:MSDOS] ; FINAL_DOS_LOCATION:0
                                                                 ;('jmp DOSINIT' in 'MSHEAD.ASM')
                                                                 ;('DOSINIT:' is in 'MSINIT.ASM')

                                    ; 22/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; 22/03/2019 - Retro DOS v4.0 (MSDOS 6.0, 6.21)

                                ; This call to DOSINIT will relocate the DOS data from its present location
                                ; at the top of memory, to its final location in low memory just above the
                                ; BIOS data. It will then build important DOS data structures in low
                                ; memory following the DOS data. It returns (among many other things) the
                                ; new starting address of free memory.

00000565 2EFF1E[7102]               call    far [cs:dosinit]      ; call dosinit
                                                    ; es:di -> sysinitvars_ext

0000056A 2E8C1E[8502]               mov     [cs:def_php],ds             ; save pointer to PSP

                                    ; 11/12/2022
                                    ; 22/03/2019
0000056F 0E                         push    cs
00000570 1F                         pop     ds
                                    ; 22/10/2022
00000571 A3[8302]                   mov     [hi_doscod_size],ax
00000574 890E[8102]                 mov     [lo_doscod_size],cx
00000578 8916[7D02]                 mov     [dos_segreinit],dx

                                    ; 11/12/2022
                                    ; ds = cs
                                    ;mov    [cs:hi_doscod_size],ax; size of doscode (including exepatch)
                                    ;mov    [cs:lo_doscod_size],cx; (not including exepatch)
                                    ;mov    [cs:dos_segreinit],dx ; save offset of segreinit

                                    ; 05/06/2018 - Retro DOS v3.0
                                    ; ES:DI = Address of pointer to SYSINITVARS structure (MSDOS 3.3)

                                    ; 11/12/2022
                                    ; ds = cs
                                    ; 22/10/2022
                                    ;mov    ax,[es:di+SysInitVars_Ext.SYSI_InitVars] ; 5/29/86
0000057C 268B05                     mov     ax,[es:di] ; 22/03/2019
                                    ;mov    [cs:DOSINFO],ax
0000057F A3[6D02]                   mov     [DOSINFO],ax
                                    ;mov    ax,[es:di+SysInitVars_Ext.SYSI_InitVars+2]
00000582 268B4502                   mov     ax,[es:di+2]
                                    ;mov    [cs:DOSINFO+2],ax
00000586 A3[6F02]                   mov     [DOSINFO+2],ax ; set the sysvar pointer

                                    ;mov    ax,[es:di+SysInitVars_Ext.SYSI_Country_Tab]
00000589 268B4504                   mov     ax,[es:di+4]
                                    ;mov    [cs:sysi_country],ax
0000058D A3[7902]                   mov     [sysi_country],ax
                                    ;mov    ax,[es:di+SysInitVars_Ext.SYSI_Country_Tab+2]
00000590 268B4506                   mov     ax,[es:di+6]
                                    ;mov    [cs:sysi_country+2],ax
00000594 A3[7B02]                   mov     [sysi_country+2],ax   ; set the SYSI_Country pointer

                                    ; 20/04/2019
                                    ;mov    ax,[CURRENT_DOS_LOCATION]
                                    ;;mov   es,[CURRENT_DOS_LOCATION]
                                    ;mov    ax,[FINAL_DOS_LOCATION] ; give dos its temporary location
                                    ; 22/10/2022
                                    ;mov    ax,[cs:CURRENT_DOS_LOCATION]
                                    ;;;mov  [dos_segreinit+2],es
                                    ;;mov   [dos_segreinit+2],ax
                                    ;mov    [cs:dos_segreinit+2],ax
                                    ; 11/12/2022
                                    ; ds = cs
00000597 8E06[7302]                 mov     es,[CURRENT_DOS_LOCATION]
0000059B 8C06[7F02]                 mov     [dos_segreinit+2],es
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
```

```
                                    ;mov    es,[cs:CURRENT_DOS_LOCATION]
                                    ;mov    [cs:dos_segreinit+2],es

                                ; ---------------------------------------------------------------------------

                                ;SYSINIT:0577h:
                                    ; ... RPLArena ... MSDOS 6.21 IO.SYS (SYSINIT:0577h to SYSINIT:05D1h)
                                ;SYSINIT:05D1h:    ; NoRPLArena

                                    ; 22/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS SYSINIT)
                                ;------ Cover up RPL code with an arena
                                ;SYSINIT:05EBh:
                                    ; 11/12/2022
                                    ; ds = cs
0000059F 31DB                       xor     bx,bx
000005A1 391E[9602]                 cmp     [RPLMemTop],bx ; 0
                                    ;cmp    word [RPLMemTop],0
                                    ;;cmp   word [cs:RPLMemTop],0
000005A5 7450                       je      short NoRPLArena

                                ;------ alloc all memory

                                    ; 11/12/2022
                                    ;mov    bx,0FFFFh
000005A7 4B                         dec     bx
                                    ; bx = 0FFFFh
000005A8 B448                       mov     ah,48h
000005AA CD21                       int     21h
                                                    ; DOS - 2+ - ALLOCATE MEMORY
                                                    ; BX = number of 16-byte paragraphs desired
000005AC B448                       mov     ah,48h
000005AE CD21                       int     21h

000005B0 8EC0                       mov     es,ax                   ; get it into ES and save it
000005B2 06                         push    es

                                ;------ resize upto RPL mem

                                    ; 11/12/2022
                                    ; ds = cs
                                    ;sub    ax,[cs:RPLMemTop]
000005B3 2B06[9602]                 sub     ax,[RPLMemTop]
000005B7 F7D8                       neg     ax
000005B9 48                         dec     ax
000005BA 89C3                       mov     bx,ax
000005BC B44A                       mov     ah,4Ah
000005BE CD21                       int     21h
                                                    ; DOS - 2+ - ADJUST MEMORY BLOCK SIZE (SETBLOCK)
                                                    ; ES = segment address of block to change
                                                    ; BX = new size in paragraphs

                                ;------ allocate the free (RPL MEM)

000005C0 BBFFFF                     mov     bx,0FFFFh
000005C3 B448                       mov     ah,48h
000005C5 CD21                       int     21h
000005C7 B448                       mov     ah,48h
000005C9 CD21                       int     21h

                                ;----- mark that it belongs to RPL

000005CB 48                         dec     ax
000005CC 8EC0                       mov     es,ax
                                    ;mov    word [es:arena_owner],8
000005CE 26C70601000800             mov     word [es:1],8
                                    ;mov    word [es:arena_name],'RP'
000005D5 26C70608005250             mov     word [es:8],'RP'
                                    ;mov    word [es:arena_name+2],'L'
000005DC 26C7060A004C00             mov     word [es:10],'L'
                                    ;mov    word [es:arena_name+4],0
000005E3 26C7060C000000             mov     word [es:12],0
                                    ;mov    word [es:arena_name+6],0
000005EA 26C7060E000000             mov     word [es:14],0

000005F1 07                             pop     es                      ; get back ptr to first block
000005F2 B449                           mov     ah,49h    ; Dealloc     ; and free it
000005F4 CD21                       int     21h
                                                                ; DOS - 2+ - FREE MEMORY
                                                                ; ES = segment address of area to be freed
                                    ; 11/12/2022
000005F6 F8                         clc

                                ; ---------------------------------------------------------------------------

                                NoRPLArena:
                                    ; 11/12/2022
                                    ; ds = cs
                                    ; 22/03/2019 - Retro DOS v4.0 (MSDOS 6.0, 6.21, IO.SYS)
000005F7 C43E[6D02]                 les     di,[DOSINFO]   ; es:di -> dosinfo
                                    ; 22/10/2022 - Retro DOS v4.0 (MSDOS 5.0 IO.SYS SYSINIT)
                                    ;les    di,[cs:DOSINFO]       ; es:di -> dosinfo

                                    ; 11/12/2022
                                    ;clc                           ; get the extended memory size

                                ;   execute the get extended memory size subfunction in the bios int 15h
                                ;   if the function reports an error do nothing else store the extended
                                ;   memory size reported at the appropriate location in the dosinfo buffer
                                ;   currently pointed to by es:di. use the offsets specified in the
                                ;   definition of the sysinitvars struct in inc\sysvar.inc

000005FB B488                       mov     ah,88h
000005FD CD15                       int     15h                     ; check extended memory size
000005FF 720B                       jc      short no_ext_memory
                                                    ; Get Extended Memory Size
                                                    ; Return: CF clear on success
                                                    ; AX = size of memory above 1M in K
                                    ;mov    [es:di+SYSI_EXT_MEM],ax ; save extended memory size
                                    ; 22/10/2022
00000601 26894545                   mov     [es:di+45h],ax ; save extended memory size
00000605 09C0                       or      ax,ax
00000607 7403                       jz      short no_ext_memory
00000609 E8F006                     call    ClrVDISKHeader
                                no_ext_memory:
                                    ;mov    ax,[es:di+SYSI_MAXSEC]; get the sector size
0000060C 268B4510                   mov     ax,[es:di+10h]
                                    ;add    ax,bufinsiz
                                    ; 30/12/2023 - Retro DOS v5.0
                                    ;add    ax,20          ; size of buffer header
00000610 83C018                     add     ax,24          ; bufinsiz
                                                           ; size of buffer header = 24 (PCDOS v7.1 IBMBIO.COM)
                                                           ; (it was 20 in MSDOS 6.22 IO.SYS)
                                    ; 11/12/2022
                                    ; ds = cs
```

```
24046 00000613 A3[9D02]                 mov     [singlebuffersize],ax ; total size for a buffer
24047                                   ;mov    [cs:singlebuffersize],ax
24048                                   ; 11/12/2022
24049 00000616 A0[9802]                 mov     al,[DEFAULT_DRIVE]    ; get the 1 based boot drive number set by msinit
24050                                   ;mov    al,[cs:DEFAULT_DRIVE]
24051                                   ;mov    [es:di+SYSI_BOOT_DRIVE],al ; set sysi_boot_drive
24052 00000619 26884543                 mov     [es:di+43h],al
24053
24054                                 ; determine if 386 system...
24055
24056                                 ; 30/12/2023
24057                                 %if 0
24058                                     ;get_cpu_type               ; macro to determine cpu type
24059
24060                                 get_cpu_type:
24061                                     ; 11/12/2022
24062                                     pushf
24063                                     ;push   bx
24064                                     ;xor    bx,bx
24065                                     ; 11/12/2022
24066                                     ;xor    cx,cx
24067                                     ;
24068                                     xor     ax,ax
24069                                     ; ax = 0
24070                                     push    ax
24071                                     popf
24072                                     pushf
24073                                     pop     ax
24074                                     and     ax,0F000h
24075                                     ;cmp    ax,0F000h
24076                                     cmp     ah,0F0h
24077                                     je      short cpu_8086
24078                                     ;mov    ax,0F000h
24079                                     mov     ah,0F0h
24080                                     ; ax = 0F000h
24081                                     push    ax
24082                                     popf
24083                                     pushf
24084                                     pop     ax
24085                                     ;and    ax,0F000h
24086                                     and     ah,0F0h
24087                                     jz      short cpu_286
24088                                 cpu_386:
24089                                     ; 11/12/2022
24090                                     ;;inc   bx
24091                                     ;inc    cx
24092                                     ; 11/12/2022
24093                                     ;mov    byte [es:di+SYSI_DWMOVE],1
24094                                     mov     byte [es:di+44h],1
24095
24096                                     ; 03/09/2023 - Retro DOS v5.0 (PCDOS 7.1 Modified SYSINIT)
24097                                     ; change A20 line on/off check code to the faster (for 32 bit cpu)
24098                                     ;push   es
24099                                     ;push   di
24100                                     ;mov    ax,DOSBIODATASEG ; 0070h
24101                                     ;mov    es,ax
24102                                     ;cld
24103                                     ;mov    di,cpu386_cmpsd ; (IsA20Off)
24104                                     ;mov    ax,4B9h         ; mov cx,4 ; B90400
24105                                     ;stosw
24106                                     ;mov    ax,0F300h       ; repz  ; F3
24107                                     ;stosw
24108                                     ;mov    ax,0A766h       ; cmpsd ; 66A7
24109                                     ;stosw
24110                                     ;pop    di
24111                                     ;pop    es
24112
24113                                 cpu_286:
24114                                     ;;;inc  bx
24115                                     ;;inc   cx
24116                                 cpu_8086:
24117                                     ; 11/12/2022
24118                                     ;;mov   ax,bx
24119                                     ;pop    bx
24120                                     popf
24121                                 %endif
24122                                     ;...
24123
24124                                     ; 11/12/2022
24125                                     ;or     cl,cl
24126                                     ;jz     short not_386_system
24127                                     ; 11/12/202
24128                                     ;cmp    cl,2
24129                                     ;;cmp   ax,2                    ; is it a 386?
24130                                     ;jne    short not_386_system  ; no: don't mess with flag
24131
24132                                     ; 30/12/2023 - Retro DOS v5.0
24133 0000061D 803E[B606]02               cmp     byte [cpu_type], 2    ; is it a 386?
24134 00000622 7505                       jne     short _not_386_cpu    ; no: don't mess with flag
24135
24136                                     ;mov    byte [es:di+SYSI_DWMOVE],1
24137                                     ; 11/12/2022
24138                                     ; 22/10/2022
24139 00000624 26C6454401                 mov     byte [es:di+44h],1
24140                                 _not_386_cpu:
24141                                     ;mov    al,[es:di+SYSI_NUMIO]
24142 00000629 268A4520                   mov     al,[es:di+20h]
24143                                     ; 11/12/2022
24144                                     ; ds = cs
24145 0000062D A2[8503]                   mov     [drivenumber],al      ; save start of installable block drvs
24146                                     ;mov    [cs:drivenumber],al
24147
24148 00000630 8CC8                       mov     ax,cs
24149 00000632 83E811                     sub     ax,11h                ; room for PSP we will copy shortly
24150                                     ; 11/12/2022
24151                                     ;mov    cx,[singlebuffersize] ; temporary single buffer area
24152                                     ;;mov   cx,[cs:singlebuffersize]
24153                                     ;shr    cx,1
24154                                     ;shr    cx,1                  ; divide size by 16...
24155                                     ;shr    cx,1
24156                                     ;shr    cx,1                  ; ...to get paragraphs...
24157                                     ;inc    cx                    ; ... and round up
24158                                     ; 11/12/2022
24159 00000635 8B1E[9D02]                 mov     bx,[singlebuffersize]
24160 00000639 B104                       mov     cl,4
24161 0000063B D3EB                       shr     bx,cl
24162 0000063D 43                         inc     bx
24163
24164                                 ;   cas note: this unorthodox paragraph rounding scheme wastes a byte
24165                                 ;     if [singlebuffersize] ever happens to be zero mod 16. Could this
24166                                 ;     ever happen? Only if the buffer overhead was zero mod 16, since
24167                                 ;     it is probably safe to assume that the sector size always will be.
24168                                 ;
24169                                 ;    mohans also found a bug in CONFIG.SYS processing where it replaces
```

```
                                ;     EOF's with cr,lf's, without checking for collision with [confbot].
                                ;     perhaps the extra byte this code guarantees is what has kept that
                                ;     other code from ever causing a problem???
                                
                                    ; 11/12/2022
0000063E 29D8                       sub     ax,bx
                                    ;sub    ax,cx
00000640 A3[A702]                   mov     [top_of_cdss],ax      ; temp "unsafe" location
                                    ; 22/10/2022
                                    ;mov    [cs:top_of_cdss],ax
                                
                                ;   chuckst -- 25 Jul 92 -- added code here to pre-allocate space
                                ;   for 26 temporary CDSs, which makes it easier to use alloclim
                                ;   for allocating memory for MagicDrv.
                                
                                    ; 30/12/2023
                                    ;push  es ; not necessary (!*)      ; preserve pointer to dosinfo
                                    ;push  di
                                
                                    ; 22/10/2022
                                ;   mov     cx,ax                 ; save pointer for buffer
                                ;
                                ;;  now allocate space for 26 CDSs
                                ;
                                ;   sub     ax,((26 *(curdirlen))+15)/16
                                ;   mov     [ALLOCLIM],ax         ; init top of free memory pointer
                                ;   mov     [CONFBOT],ax          ; init this in case no CONFIG.SYS
                                
                                    ; 30/12/2022 - Retro DOS v4.2 (MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:064Ch)
00000643 89C1                       mov     cx,ax ; (*)
00000645 2D8F00                     sub     ax,((26*(curdirlen))+15)/16 ; sub ax,143
00000648 A3[A502]                   mov     [ALLOCLIM],ax         ; init top of free memory pointer
0000064B A3[A302]                   mov     [CONFBOT],ax          ; init this in case no CONFIG.SYS
                                
                                ; setup and initialize the temporary buffer at cx
                                
                                    ;les    di,[es:di+SYSI_BUF]   ; get the buffer chain entry pointer
0000064E 26C47D12                   les     di,[es:di+12h]
                                    ; 11/12/2022
00000652 31DB                       xor     bx,bx
                                    ;xor    ax,ax
                                    ;mov    [es:di+BUFFINF.Dirty_Buff_Count],ax ; 0
                                    ;mov    word [es:di+4],0
00000654 26895D04                   mov     [es:di+4],bx ; 0
                                    ;mov    [es:di+BUFFINF.Buff_Queue],ax ; 0
                                    ;mov    word [es:di],0
00000658 26891D                     mov     [es:di],bx ; 0
                                    ;mov    [es:di+BUFFINF.Buff_Queue+2],cx ; cx = [top_of_cdss] ; 6.21
                                    ;;mov   [es:di+BUFFINF.Buff_Queue+2],ax ; ax = [top_of_cdss] ; 5.0
                                    ;mov    [es:di+2],ax
                                    ;mov    es,ax  ; [top_of_cdss] = [CONFBOT]
                                    ; 30/12/2022 - Retro DOS v4.2 (MSDOS 6.21 IO.SYS, SYSINIT)
0000065B 26894D02                   mov     [es:di+2],cx ; [top_of_cdss] ; (*)
0000065F 8EC1                       mov     es,cx
                                
                                    ; 11/12/2022
                                    ;xor    ax,ax
                                    ;mov    di,ax                 ; es:di -> single buffer
00000661 89DF                       mov     di,bx
                                    ; di = 0
                                
                                    ;mov    [es:di+buffinfo.buf_next],ax ; points to itself
                                    ; 11/12/2022
                                    ;mov    [es:di],ax ; 0
00000663 26891D                     mov     [es:di],bx ; 0
                                    ;mov    [es:di+buffinfo.buf_prev],ax ; points to itself
                                    ; 11/12/2022
                                    ;mov    [es:di+2],ax ; 0
00000666 26895D02                   mov     [es:di+2],bx ; 0
                                
                                    ; 22/10/2022 - Retro DOS 4.0 (Modified MSDOS 5.0 IO.SYS SYINIT)
                                    ; MSDOS 5.0 IO.SYS - SYSINIT:06E0h
                                
                                    ;mov    word [es:di+buffinfo.buf_ID],00FFh ; free buffer,clear flag
0000066A 26C74504FF00               mov     word [es:di+4],00FFh
                                ;SYSINIT:06E6h
                                    ;;mov   [es:di+buffinfo.buf_sector],ax ; 0
                                    ;mov    word [es:di+6],0
                                    ; 11/12/2022
                                    ;mov    [es:di+buffinfo.buf_sector],bx ; 0
00000670 26895D06                   mov     [es:di+6],bx ; 0
                                    ;;mov   [es:di+buffinfo.buf_sector+2],ax ; 0
                                    ;mov    word [es:di+8],0
                                    ; 11/12/2022
                                    ;mov    [es:di+buffinfo.buf_sector+2],bx ; 0
00000674 26895D08                   mov     [es:di+8],bx ; 0
                                
                                    ; 30/12/2023 (!*)
                                    ;pop    di                    ; restore pointer to DOSINFO data
                                    ;pop    es
                                
                                    ; 11/12/2022
                                    ; ds = cs
                                    ; 22/10/2022
                                    ;push   cs
                                    ;pop    ds
                                
00000678 E82807                     call    TempCDS               ; set up cdss so re_init and sysinit
                                                                  ;  can make disk system calls
                                                                  ; tempcds trashes ds
                                    ; 10/05/2019
0000067B 2E8E1E[8502]               mov     ds,[cs:def_php]             ; retrieve pointer to PSP returned by DOSINIT
                                
                                    ;if not ibmjapver
                                    ;call   far KERNEL_SEGMENT:re_init ; re-call the bios
                                    ;endif
                                
                                    ; 22/10/2022
                                ;SYSINIT:06FEh:    ; (MSDOS 5.0 IO.SYS, SYSINIT)
                                    ; 30/12/2022
                                ;SYSINIT:0697h:    ; (MSDOS 6.21 IO.SYS, SYSINIT)
                                    ;call   far ptr 70h:89Bh
00000680 9A[2807]7000               call    DOSBIODATASEG:RE_INIT
                                
00000685 FB                         sti                           ; ints ok
00000686 FC                         cld                           ; make sure
                                
                                ; 23/03/2019
                                
                                ;SYSINIT:069Eh     ; 30/12/2022
                                
                                ; dosinit has set up a default "process" (php) at ds:0. we will move it out
                                ; of the way by putting it just below sysinit at end of memory.
```

```
24294
24295 00000687 8CCB                    mov     bx,cs
24296 00000689 83EB10                  sub     bx,10h
24297 0000068C 8EC3                    mov     es,bx
24298 0000068E 31F6                    xor     si,si
24299 00000690 89F7                    mov     di,si
24300 00000692 B98000                  mov     cx,128
24301 00000695 F3A5                    rep     movsw
24302
24303                                  ;mov    [es:PDB.JFN_POINTER+2],es ; Relocate
24304                                  ; 22/10/2022
24305 00000697 268C063600              mov     [es:36h],es
24306
24307                                  ; Set Process Data Block - Program Segment Prefix address
24308                                  ; BX = PDB/PSP segment
24309 0000069C B450                      mov ah,50h  ; SET_CURRENT_PDB
24310 0000069E CD21                    int     21h                     ; tell DOS we moved it
24311                                                  ; DOS - 2+ internal - SET PSP SEGMENT
24312                                                  ; BX = segment address of new PSP
24313                                  ; 22/10/2022
24314                                  ; 27/03/2019
24315                                  ; 30/12/2023
24316                                  ;push   ds ; */                 ; preserve DS returned by DOSINIT
24317
24318 000006A0 0E                      push    cs
24319 000006A1 1F                      pop     ds
24320
24321                                  ; set up temp. critical error handler
24322 000006A2 BA[794A]                mov     dx,int24                ; set up int 24 handler
24323                                  ;;mov    ax,(SET_INTERRUPT_VECTOR*256)+24h
24324                                  ;mov    ax,(SET_INTERRUPT_VECTOR<<8)|24h
24325 000006A5 B82425                  mov     ax,2524h
24326 000006A8 CD21                    int     21h
24327
24328 000006AA 803E[8803]00                cmp     byte [toomanydrivesflag],0 ; Q: >24 partitions?      M029
24329 000006AF 7406                        je      short no_err               ;  N: continue            M029
24330 000006B1 BA[9E53]                    mov     dx,TooManyDrivesMsg        ;  Y: print error message M029
24331                                      ; 22/10/2022
24332                                      ;call   print                      ;                         M029
24333                                      ; 12/12/2022
24334 000006B4 EB04                        jmp     short p_dosinit_msg ; 23/03/2019 - Retro DOS v4.0
24335
24336                                      ; 30/12/2023 - Retro DOS v5.0
24337                                  cpu_type:
24338 000006B6 FF                          db 0FFh ; db 0
24339
24340                                  no_err:
24341                                      ; 12/05/2019
24342                                      ;-----------------------------------------------
24343                                      ; 27/06/2018 - Retro DOS v3.0 ; 23/03/2019 - Retro DOS v4.0
24344                                      ; 22/10/2022 - Retro DOS v4.0
24345                                      ; 12/12/2022
24346                                      ; 30/12/2023 - Retro DOS v5.0
24347 000006B7 BA[7D4A]                    mov     dx,BOOTMES              ; Display (fake) MSDOS version message
24348                                  p_dosinit_msg:
24349 000006BA E89743                      call    print                   ; Print message
24350                                      ;-----------------------------------------------
24351
24352                                      ; 11/12/2022
24353                                      ; 22/10/2022
24354                                      ; 23/03/2019 - Retro DOS v4.0
24355                                      ;pop    ds                      ; start of free memory
24356                                      ;mov    dl,[cs:DEFAULT_DRIVE]
24357
24358                                      ; 11/12/2022
24359                                      ; 27/03/2019
24360 000006BD 8A16[9802]                  mov     dl,[DEFAULT_DRIVE]
24361                                      ; 30/12/2023
24362                                      ;pop    ds ; */
24363
24364 000006C1 08D2                        or      dl,dl
24365                                      ; 30/12/2023
24366 000006C3 7405                        jz      short nodrvset          ; bios didn't say
24367                                      ;jz     short ProcessConfig  ; (Retro DOS v4.0 does not contain DBLSPACE code)
24368                                      ;dec    dl                      ; A = 0
24369                                      ; 18/12/2022
24370 000006C5 4A                          dec     dx
24371 000006C6 B40E                        mov     ah,0Eh  ; SET_DEFAULT_DRIVE
24372 000006C8 CD21                        int     21h                     ; select the disk
24373                                                    ; DOS - SELECT DISK
24374                                                    ; DL = new default drive number (0 = A, 1 = B, etc.)
24375                                                    ; Return: AL = number of logical drives
24376                                  nodrvset:
24377                                      ; 04/01/2023 - Retro DOS 4.2 (Modified MSDOS 6.21 IO.SYS SYINIT)
24378                                      ; (SYSINIT:06DFh)
24379
24380                                      ; 30/12/2023 - Retro DOS 5.0
24381                                      ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:0733h)
24382 000006CA 1E                          push    ds
24383 000006CB 29C0                        sub     ax,ax
24384 000006CD 8ED8                        mov     ds,ax ; 0 ; ROM BIOS Data Area
24385 000006CF A16C04                      mov     ax,[46Ch] ; timer tick count (18.2 ticks per second)
24386                                      ;mov    [cs:_timer_lw_],ax
24387 000006D2 1F                          pop     ds
24388                                      ; ds = cs
24389 000006D3 A3[8C03]                    mov     [_timer_lw_],ax
24390
24391                                      ; ---------------------
24392
24393                                      ;ifdef dblspace_hooks
24394                                      ;       ....
24395                                      ;       ....
24396                                      ;endif
24397
24398                                      ; ---------------------
24399
24400                                      ; 30/12/2023 - Retro DOS 5.0 (Modified MSDOS 7.1 IBMBIO.COM SYS SYINIT)
24401                                  ; ---------------------------------------------------------------------------
24402                                      ; PCDOS 7.1 IBMBIO.COM - SYSINIT:0740h
24403
24404 000006D6 0E                          push    cs
24405 000006D7 07                          pop     es
24406
24407                                      ; 07/04/2024
24408                                      ;mov    word [cs:MagicBackdoor+2],cs
24409                                      ;mov    word [cs:MagicBackdoor], NullBackdoor
24410 000006D8 8C0E[9203]                  mov     word [MagicBackdoor+2],cs
24411 000006DC C706[9003][9403]            mov     word [MagicBackdoor], NullBackdoor
24412
24413                                      ; ds = es = cs = SYSINIT segment
24414                                  set_drvspc_size:
24415 000006E2 BE[AB16]                    mov     si,MagicDDName          ; "\DBLSPACE.BIN"
24416                                  set_dblspc_size:
24417 000006E5 E8792F                      call    SizeDevice
```

```
24418 000006E8 732B                     jnc     short wait_for_key_2s
24419                                   ;cmp    byte [cs:si], 'C'
24420 000006EA 803C43                   cmp     byte [si],'C'           ; "C:\STACKER.BIN"
24421 000006ED 740C                     je      short set_drvspc_name
24422                                   ;cmp    byte [cs:DEFAULT_DRIVE],3
24423 000006EF 803E[9802]03             cmp     byte [DEFAULT_DRIVE],3
24424 000006F4 7405                     je      short set_drvspc_name
24425 000006F6 83EE02                   sub     si,2                    ; "C:\DBLSPACE.BIN"
24426 000006F9 EBEA                     jmp     short set_dblspc_size
24427
24428                                 set_drvspc_name:
24429                                   ;cmp    byte [cs:MagicDDName+2],'R'   ; "BLSPACE.BIN"
24430 000006FB 803E[AD16]52             cmp     byte [MagicDDName+2],'R'
24431 00000700 7408                     je      short set_stacker_name
24432                                   ;mov    word [cs:MagicDDName+2],'RV'  ; "DRVSPACE.BIN"
24433 00000702 C706[AD16]5256           mov     word [MagicDDName+2],'RV'
24434 00000708 EBD8                     jmp     short set_drvspc_size
24435
24436                                 set_stacker_name:
24437 0000070A 81FE[B916]               cmp     si,StackerName          ; "C:\STACKER.BIN"
24438 0000070E 734B                     jnb     short wfk2s_4
24439 00000710 BE[BB16]                 mov     si,StackerName+2        ; "\STACKER.BIN"
24440 00000713 EBD0                     jmp     short set_dblspc_size
24441
24442                                 wait_for_key_2s:
24443                                   ;mov    [cs:MagicDDNamePtr],si
24444 00000715 8936[A716]               mov     [MagicDDNamePtr],si
24445 00000719 1E                       push    ds
24446 0000071A 29C0                     sub     ax,ax
24447 0000071C 8ED8                     mov     ds,ax   ; 0             ; ROMBIOS data area
24448 0000071E 8B166C04                 mov     dx,[46Ch]               ; Counter for Interrupt 1Ah
24449                                 wfk2s_1:
24450 00000722 B401                     mov     ah,1
24451 00000724 CD16                     int     16h             ; KEYBOARD - CHECK BUFFER, DO NOT CLEAR
24452                                                           ; Return: ZF clear if character in buffer
24453                                                           ; AH = scan code, AL = character
24454                                                           ; ZF set if no character in buffer
24455 00000726 7511                     jnz     short wfk2s_2
24456 00000728 B402                     mov     ah,2
24457 0000072A CD16                     int     16h                     ; KEYBOARD - GET SHIFT STATUS
24458                                                                   ; AL = shift status bits
24459 0000072C A803                     test    al,3
24460 0000072E 7509                     jnz     short wfk2s_2
24461 00000730 A16C04                   mov     ax,[46Ch]               ; tick count
24462 00000733 29D0                     sub     ax,dx
24463 00000735 3C25                         cmp al,37                   ; 2 seconds
24464 00000737 72E9                     jb      short wfk2s_1           ; wait for user's key press
24465                                 wfk2s_2:
24466 00000739 1F                       pop     ds                      ; read/check the pressed key
24467 0000073A 29DB                     sub     bx,bx   ; bx = 0
24468 0000073C B402                     mov     ah,2
24469 0000073E CD16                     int     16h                     ; KEYBOARD - GET SHIFT STATUS
24470                                                                   ; AL = shift status bits
24471 00000740 A803                     test    al,3                    ; Left or Right SHIFT key pressed ?
24472 00000742 7402                     jz      short wfk2s_3           ; no
24473 00000744 43                       inc     bx
24474 00000745 43                       inc     bx      ; bx = 2
24475                                 wfk2s_3:
24476 00000746 B401                     mov     ah,1
24477 00000748 CD16                     int     16h             ; KEYBOARD - CHECK BUFFER, DO NOT CLEAR
24478                                                           ; Return: ZF clear if character in buffer
24479                                                           ; AH = scan code, AL = character
24480                                                           ; ZF set if no character in buffer
24481 0000074A 7418                     jz      short wfk2s_6
24482 0000074C 80FC65                   cmp     ah,65h                  ; F8 key pressed ?
24483 0000074F 740C                     jz      short wfk2s_5
24484 00000751 80FC62                   cmp     ah,62h                  ; F5 key pressed ?
24485 00000754 750E                     jnz     short wfk2s_6
24486                                   ;mov    byte [cs:F5_key],1
24487 00000756 C606[8E03]01             mov     byte [F5_key],1
24488                                 wfk2s_4:
24489 0000075B EB49                     jmp     short ProcessConfig     ; continue (as normal/default state)
24490
24491                                 wfk2s_5:
24492                                   ;mov    byte [cs:F8_key],1
24493 0000075D C606[8F03]01             mov     byte [F8_key],1
24494 00000762 EB42                     jmp     short ProcessConfig
24495
24496                                 wfk2s_6:
24497 00000764 E8AA02                   call    AllocFreeMem            ; get the largest free block from DOS
24498 00000767 E8700F                   call    MagicPreload            ; **** PRE-LOAD MAGICDRV!!! ****
24499
24500                                   ; 07/04/2024 - Retro DOS v5.0
24501                                   ; (DS may not be same with CS here!)
24502 0000076A 0E                       push    cs
24503 0000076B 1F                       pop     ds ; *
24504 0000076C 8E06[6803]               mov     es,[area]
24505
24506 00000770 09C0                     or      ax,ax                   ; error?
24507 00000772 7406                     jz      short wfk2s_7
24508                                 PreloadFailed:
24509 00000774 B449                     mov     ah,49h                  ; Dealloc ; free the block if no load
24510                                   ;;mov   es,[cs:area]
24511                                   ;mov    es,[area]
24512 00000776 CD21                     int     21h             ; DOS - 2+ - FREE MEMORY
24513                                                           ; ES = segment address of area to be freed
24514 00000778 EB2C                     jmp     short ProcessConfig
24515
24516                                 wfk2s_7:
24517                                   ;mov    bx,[cs:memhi]
24518                                   ;mov    es,[cs:area]
24519                                   ;sub    bx,[cs:area]            ; get desired block size in paras
24520                                   ; 07/04/2024 - Retro DOS v5.0
24521                                   ; ds = cs ; *
24522 0000077A 8CC3                     mov     bx,es
24523 0000077C F7DB                     neg     bx              ; bx = - [cs:area]
24524 0000077E 031E[6403]               add     bx,[memhi]      ; bx = [cs:memhi] - [cs:area]
24525
24526 00000782 B44A                     mov     ah,4Ah
24527 00000784 CD21                     int     21h             ; DOS - 2+ - ADJUST MEMORY BLOCK SIZE (SETBLOCK)
24528                                                           ; ES = segment address of block to change
24529                                                           ; BX = new size in paragraphs
24530 00000786 8CC0                     mov     ax,es
24531 00000788 48                       dec     ax
24532 00000789 8EC0                     mov     es,ax                   ; get Magicdrv arena
24533
24534 0000078B 26C70601000800           mov     word [es:1],8  ; [es:arena_owner]
24535                                   ;mov    word [es:ARENA.OWNER],8       ; set impossible owner
24536 00000792 26C70608005344           mov     word [es:8],4453h ; [es:arena_name],'SD' ; System Data
24537                                   ;mov    word [es:ARENA.NAME],'SD' ; 4453h
24538 00000799 2603060300                   add ax,[es:3]               ; get MCB length
24539                                   ;add    ax,[es:ARENA.SIZE]
24540
24541                                   ;lds    si,[cs:DOSINFO]                 ; get to arena header
```

```
24542 0000079E C536[6D02]                   lds     si,[DOSINFO]
24543 000007A2 40                               inc     ax              ; get addr of next MCB
24544 000007A3 8944FE                           mov     [si-2], ax      ; store that
24545
24546                                   ; ----------------------------------------------------------------------------
24547
24548                                   ; MSDOS 6.21 IO.SYS, SYSINIT:0744h
24549
24550                                   ; 23/03/2019 - Retro DOS v4.0 (MSDOS 6.0, SYSINIT1.ASM, 1991)
24551                                   ; ----------------------------------------------------------------------------
24552                                   ; 22/10/2022 - Retro DOS v4.0 (MSDOS 5.0 IO.SYS SYSINIT)
24553                                   ; ----------------------------------------------------------------------------
24554                                   ; 30/12/2022 - Retro DOS v4.2 (MSDOS 6.21 IO.SYS SYSINIT)
24555                                   ; ----------------------------------------------------------------------------
24556                                   ; 07/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM SYSINIT)
24557
24558                                   ; (MSDOS 6.22 IO.SYS - SYSINIT:0744h)
24559
24560                                   ; PCDOS 7.1 IBMBIO.COM - SYSINIT:0820h
24561
24562                                   ProcessConfig:
24563                                       ;; ds = cs ; 27/03/2019
24564                                       ; 11/12/2022
24565                                       ; ds <> cs
24566
24567                                   ; (MSDOS 5.0 IO.SYS - SYSINIT:0746h)
24568
24569 000007A6 E8BF1C                       call    doconf                  ; do pre-scan for dos=high/low
24570
24571                                       ; 11/12/2022
24572                                       ; 27/03/2019
24573                                       ; ds = cs (at return from doconf)
24574
24575                                   ; Now, if this is not romdos, we decide what to do with the DOS code.
24576                                   ; It will either be relocated to low memory, above the DOS data structures,
24577                                   ; or else it will be located in HiMem, in which case a stub with the DOS
24578                                   ; code entry points will be located in low memory. Dos_segreinit is used
24579                                   ; to tell the DOS data where the code has been placed, and to install the
24580                                   ; low memory stub if necessary. If the DOS is going to go into HiMem, we
24581                                   ; must first initialize it in its present location and load the installable
24582                                   ; device drivers. Then, if a HiMem driver has been located, we can actually
24583                                   ; relocate the DOS code into HiMem.
24584                                   ;
24585                                   ; For ROMDOS, if DOS=HIGH is indicated, then we need to call dos_segreinit
24586                                   ; to install the low memory stub (this must be done before allowing any
24587                                   ; device drivers to hook interrupt vectors). Otherwise, we don't need to
24588                                   ; call dos_segreinit at all, since the interrupt vector table has already
24589                                   ; been patched.
24590
24591                                       ; 22/10/2022 - Retro DOS v4.0
24592                                       ; (MSDOS 5.0 IO.SYS - SYSINIT:0749h)
24593                                       ;cmp    byte [cs:runhigh],0   ; Did user choose to run low ?
24594                                       ; 11/12/2022
24595 000007A9 803E[6C02]00                cmp     byte [runhigh],0
24596 000007AE 7404                        je      short dont_install_stub       ; yes, don't install dos low mem stub
24597
24598                                   ;------ user chose to load high
24599
24600                                       ; 22/10/2022
24601                                       ;mov    es,[cs:CURRENT_DOS_LOCATION] ; MSDOS 6.21 (& MSDOS 6.0)
24602                                       ; 11/12/2022
24603                                       ; ds = cs
24604                                   ; 13/04/2024
24605                                   %if 0
24606                                       mov     es,[CURRENT_DOS_LOCATION]
24607                                   %endif
24608                                       ;mov    es,[cs:FINAL_DOS_LOCATION] ; Retro DOS v4.0
24609                                       ; 27/03/2019
24610                                       ;;mov   es,[FINAL_DOS_LOCATION]
24611
24612 000007B0 31C0                        xor     ax,ax                   ; ax = 0 ---> install stub
24613
24614                                   ; 13/04/2024
24615                                   %if 0
24616                                       ; 11/12/2022
24617                                       ; ds = cs
24618                                       ;call   far [cs:dos_segreinit]; call dos segreinit
24619                                       call    far [dos_segreinit]
24620                                   %endif
24621 000007B2 EB08                        jmp     short do_multi_pass
24622
24623                                   ;------ User chose to load dos low
24624
24625                                   dont_install_stub:
24626                                       ; 22/10/2022
24627 000007B4 31DB                        xor     bx,bx                   ; M012
24628                                                                       ; don't use int 21 call to alloc mem
24629 000007B6 E80E03                      call    MovDOSLo                ; move it !
24630
24631 000007B9 B80100                      mov     ax,1                    ; dont install stub
24632
24633                                   ; 13/04/2024
24634                                   %if 1
24635                                   do_multi_pass:
24636                                   %endif
24637                                       ; 11/12/2022
24638                                       ; ds = cs
24639 000007BC 8E06[7302]                  mov     es,[CURRENT_DOS_LOCATION]
24640                                       ;mov    es,[cs:CURRENT_DOS_LOCATION] ; set_dos_final_position set it up
24641                                       ;;mov   es,[cs:FINAL_DOS_LOCATION]   ; Retro DOS v4.0
24642                                       ; 27/03/2019
24643                                   ;do_multi_pass:
24644                                       ;;mov   es,[FINAL_DOS_LOCATION]
24645
24646                                       ; 11/12/2022
24647                                       ; ds =cs
24648                                       ;call   far [cs:dos_segreinit]; inform dos about new seg
24649 000007C0 FF1E[7D02]                  call    far [dos_segreinit]
24650
24651                                   ; 13/04/2024
24652                                   %if 0
24653                                   do_multi_pass:
24654                                   %endif
24655
24656 000007C4 E84A02                      call    AllocFreeMem            ; allocate all the free mem
24657                                                                       ; & update [memhi] & [area]
24658                                                                       ; start of free memory.
24659                                       ;ifdef dblspace_hooks
24660                                       ;mov    bx,0                    ; magic backdoor to place int hooks
24661                                       ;call   cs:MagicBackdoor
24662                                       ;endif
24663
24664                                       ; 07/04/2024 - Retro DOS v5.0
24665                                       ; (PCDOS 7.1 IBMBIO.COM)
```

```
                                    ;cmp    byte [cs:F5_key],1
000007C7 803E[8E03]01               cmp     byte [F5_key],1
000007CC 740D                       je      short skip_magicbackdoor
                                    ;cmp    byte [cs:F8_key],1
000007CE 803E[8F03]01               cmp     byte [F8_key],1
000007D3 7406                       je      short skip_magicbackdoor
000007D5 31DB                       xor     bx,bx   ; bx = 0        ; magic backdoor to place int hooks
                                    ;call   far [cs:MagicBackdoor]
000007D7 FF1E[9003]                 call    far [MagicBackdoor]

                                skip_magicbackdoor:

                                ; Now, process config.sys some more.
                                ; Load the device drivers and install programs

                                    ; 22/10/2022
                                    ;inc    byte [cs:multi_pass_id]      ; multi_pass_id = 1
                                    ; 11/12/2022
                                    ; ds = cs
000007DB FE06[CD02]                 inc     byte [multi_pass_id]
000007DF E8221D                     call    multi_pass              ; load device drivers
000007E2 E8EA31                     call    ShrinkUMB
000007E5 E80E32                     call    UnlinkUMB               ; unlink all UMBs     ;M002
                                    ; 02/11/2022
                                    ;inc    byte [cs:multi_pass_id]      ; multi_pass_id = 2
                                    ; 11/12/2022
                                    ; ds = cs
000007E8 FE06[CD02]                 inc     byte [multi_pass_id]
000007EC E8151D                     call    multi_pass              ; was load ifs (now does nothing)

                                    ;ifdef dblspace_hooks
                                    ;call   MagicPostload           ; make sure Magicdrv is final placed
                                    ;endif

                                    ; ds = cs

                                    ; 07/04/2024
                                    ;call   endfile                 ; setup fcbs, files, buffers etc

                                    ;ifdef dblspace_hooks
                                    ;call   MagicSetCdss            ; disable CDSs of reserved drives
                                    ;endif

                                    ; 07/04/2024 - Retro DOS v5.0
                                    ; (PCDOS 7.1 IBMBIO.COM)
                                    ;cmp    byte [cs:F5_key],1
000007EF 803E[8E03]01               cmp     byte [F5_key],1
000007F4 7412                       je      short skip_magicpostload
                                    ;cmp    byte [cs:F8_key],1
000007F6 803E[8F03]01               cmp     byte [F8_key],1
000007FB 740B                       je      short skip_magicpostload
000007FD E8B710                     call    MagicPostload           ; make sure Magicdrv is final placed
                                    ; 13/04/2024
                                    ; ds = cs
00000800 E83E06                     call    endfile                 ; setup fcbs, files, buffers etc
00000803 E81011                     call    MagicSetCdss            ; disable CDSs of reserved drives
                                    ; ds = cs
00000806 EB03                       jmp     short _@_

                                skip_magicpostload:
                                    ; 13/04/2024
                                    ; ds = cs
00000808 E83606                     call    endfile                 ; setup fcbs, files, buffers etc
                                _@_:

                                ;Reset SysinitPresent flag here. This is needed for the special fix for lying
                                ;to device drivers. This has been moved up to this point to avoid problems
                                ;with overlays called from installed programs

                                    ; 11/12/2022
                                    ; ds = cs

                                    ;;mov   ax,Bios_Data ; 0070h
                                    ;mov    ax,KERNEL_SEGMENT
                                    ; 21/10/2022
0000080B B87000                     mov     ax,DOSBIODATASEG ; 0070h
0000080E 8EC0                       mov     es,ax                   ; point ES to bios data

00000810 26C606[DD07]00             mov     byte [es:SysinitPresent],0 ; clear SysinitPresent flag

                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;test   word [cs:install_flag],have_install_cmd ; 1
                                    ;test   byte [cs:install_flag],1
                                    ; 11/12/2022
                                    ; ds = cs
00000816 F606[CE02]01               test    byte [install_flag],1
                                    ;test   byte [cs:install_flag],have_install_cmd
                                                                    ; are there install commands?
0000081B 7407                       jz      short dolast            ; no, no need for further processing
                                    ;inc    byte [cs:multi_pass_id]      ; mult_pass_id = 3
                                    ; 11/12/2022
                                    ; ds =cs
0000081D FE06[CD02]                 inc     byte [multi_pass_id]
00000821 E8E01C                     call    multi_pass              ; execute install= commands

                                dolast:

                                ; [area] has the segment address for the allocated memory of sysinit, confbot.
                                ;  free the confbot area used for config.sys and sysinit itself.

                                ; Now if DOS is supposed to run high, we actually move it into high memory
                                ; (if HiMem manager is available). For ROMDOS, we don't actually move
                                ; anything, but just set up the ROM area for suballocation (or print
                                ; a message if HiMem is not available).
                                ;
                                ; There is also this little hack for CPM style DOS calls that needs to
                                ; be done when A20 is set...

                                    ; 11/12/2022
                                    ; ds = cs

                                    ; 22/10/2022
                                    ;cmp    byte [cs:runhigh],0FFh; are we still waiting to be moved?
                                    ; 11/12/2022
00000824 803E[6C02]FF               cmp     byte [runhigh],0FFh
00000829 7503                       jne     short _@@_ ; 09/12/2022 ; no, our job is over
0000082B E84802                     call    LoadDOSHiOrLo
                                _@@_:
                                    ;cmp    byte [cs:runhigh],0   ; are we running low
                                    ; 11/12/2022
                                    ; ds = cs
0000082E 803E[6C02]00               cmp     byte [runhigh],0
                                    ;je     short _@@@
00000833 7403                       je      short ConfigDone        ; yes, no CPM hack needed
```

```
24790 00000835 E84C05                  call    CPMHack                 ; make ffff:d0 same as 0:c0
24791                                  _@@@:
24792
24793                                  ; We are now done with CONFIG.SYS processing
24794
24795                                  ConfigDone:
24796                                      ; 12/12/2022
24797                                      ; 22/10/2022
24798                                      ;mov    byte [cs:donotshownum],1
24799                                                                      ; done with config.sys.
24800                                                                      ; do not show line number message.
24801                                      ;mov    es,[cs:area]
24802                                      ; 12/12/2022
24803                                      ; ds = cs
24804                                      ; 27/03/2019
24805 00000838 C606[5503]01              mov     byte [donotshownum],1
24806 0000083D 8E06[6803]                mov     es,[area]
24807
24808 00000841 B449                          mov     ah,49h ; DEALLOC  ; free allocated memory for command.com
24809 00000843 CD21                      int     21h
24810                                                    ; DOS - 2+ - FREE MEMORY
24811                                                    ; ES = segment address of area to be freed
24812
24813                                      ; 22/10/2022
24814                                      ;test   word [cs:install_flag],2
24815                                      ;test   word [cs:install_flag],has_installed ; sysinit_base installed?
24816                                      ;test   byte [cs:install_flag],has_installed
24817                                      ; 11/12/2022
24818                                      ; ds = cs
24819 00000845 F606[CE02]02              test    byte [install_flag],2 ; has_installed
24820                                      ;test   byte [install_flag],has_installed
24821 0000084A 741F                      jz      short skip_free_sysinitbase ; no.
24822
24823                                  ; set block from the old_area with impossible_owner_size.
24824                                  ; this will free the unnecessary sysinit_base that had been put in memory to
24825                                  ; handle install= command.
24826
24827                                      ; 12/12/2022
24828                                          ;push      es              ; BUGBUG 3-30-92 JeffPar: no reason to save ES
24829                                      ;push   bx
24830
24831                                      ; 22/10/2022
24832                                      ;mov    es,[cs:old_area]
24833                                      ;mov    bx,[cs:impossible_owner_size]
24834                                      ; 12/12/2022
24835                                      ; ds = cs
24836 0000084C 8E06[5E03]                mov     es,[old_area]
24837 00000850 8B1E[6003]                mov     bx,[impossible_owner_size]
24838
24839 00000854 B44A                      mov     ah,4Ah ; SETBLOCK
24840 00000856 CD21                      int     21h
24841                                                    ; DOS - 2+ - ADJUST MEMORY BLOCK SIZE (SETBLOCK)
24842                                                    ; ES = segment address of block to change
24843                                                    ; BX = new size in paragraphs
24844 00000858 8CC0                      mov     ax,es
24845 0000085A 48                        dec     ax
24846 0000085B 8EC0                      mov     es,ax                   ; point to arena
24847                                    ;mov    word [es:ARENA.OWNER],8       ; set impossible owner
24848 0000085D 26C70601000800            mov     word [es:1],8
24849                                    ;mov    word [es:ARENA.NAME],'SD' ; 4453h ; System Data
24850 00000864 26C70608005344            mov     word [es:8],'SD'
24851
24852                                      ; 12/12/2022
24853                                      ;pop    bx
24854                                          ;pop       es              ; BUGBUG 3-30-92 JeffPar: no reason to save ES
24855
24856                                  skip_free_sysinitbase:
24857                                      ; 22/10/2022
24858                                      ;cmp    byte [cs:runhigh],0
24859                                      ; 12/12/2022
24860                                      ; ds = cs
24861 0000086B 803E[6C02]00              cmp     byte [runhigh],0
24862 00000870 7403                      je      short _@@@_ ; 04/07/2023
24863
24864 00000872 E8DF03                    call    InstVDiskHeader         ; Install VDISK header (allocates some mem from DOS)
24865
24866                                  ; ----------------------------------------------------------------------------
24867
24868                                  _@@@_:
24869                                      ; 12/12/2022
24870                                      ; ds = cs
24871                                      ; 22/10/2022
24872                                      ; 27/03/2019
24873                                      ;push   cs
24874                                      ;pop    ds                      ; point DS to sysinitseg
24875
24876                                  ; set up the parameters for command
24877
24878                                  ;   ; 22/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS SYSINIT)
24879                                  ;;ifdef     MULTI_CONFIG
24880                                  ;   mov     byte [config_cmd],0   ; set special code for query_user
24881                                  ;       call    query_user        ; to issue the AUTOEXEC prompt
24882                                  ;   jnc     short process_autoexec; we should process autoexec normally
24883                                  ;   ; !!!
24884                                  ;   or      byte [bQueryOpt],4 ; MSDOS 6.21 IO.SYS - SYSINIT:081Fh
24885                                  ;       ; !!!
24886                                  ;   call    disable_autoexec        ; no, we should disable it
24887                                  ;process_autoexec:
24888                                  ;;endif     ; !!!
24889                                  ;   call    CheckQueryOpt  ; MSDOS 6.21 IO.SYS - SYSINIT:0827h
24890                                  ;   ; !!!
24891
24892                                      ; 22/10/2022
24893                                      ;mov     cl,[command_line]
24894                                          ;mov     ch,0
24895                                          ;inc     cx
24896                                          ;mov     si,command_line
24897                                      ;add     si,cx
24898                                          ;mov     byte [si],cr     ; cr-terminate command line
24899
24900                                      ; 22/10/2022 - Retro DOS 4.0 (Modified MSDOS 5.0 IO.SYS)
24901                                      ; (SYSINIT:0809h)
24902
24903                                      ;;;;
24904
24905                                      ; 30/12/2022 - Retro DOS 4.2 (Modified MSDOS 6.21 IO.SYS)
24906                                      ; (SYSINIT:0813h)
24907                                      ; ds = cs
24908                                      ; push  cs
24909                                      ; pop   ds
24910
24911 00000875 C606[6419]00              mov     byte [config_cmd],0   ; set special code for query_user
24912 0000087A E89F3D                    call    query_user            ; to issue the AUTOEXEC prompt
24913                                      ; 07/04/2024
```

```
                                    ;jnc    short process_autoexec; we should process autoexec normally
                                    ; 07/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                    ;;;
0000087D 9C                         pushf
0000087E F606[814C]01               test    byte [bDisableUI],1
00000883 7507                       jnz     short _@@@@_   ; F5 clean/interactive boot option (has been) disabled
00000885 803E[8E03]01               cmp     byte [F5_key],1
0000088A 7405                       je      short _@@@@@_  ; F5 key pressed, bypass AUTOEXEC.BAT (clean boot)
                                _@@@@_:
0000088C 9D                         popf
0000088D 730B                       jnc     short process_autoexec; we should process autoexec normally
0000088F EB01                       jmp     short bypass_autoexec
                                _@@@@@_:
00000891 9D                         popf                    ; cf status at the return from 'query_user' call
                                bypass_autoexec:
                                    ;;;
                                    ; !!!
00000892 800E[854C]04               or      byte [bQueryOpt],4 ; MSDOS 6.21 IO.SYS - SYSINIT:081Fh
                                    ; !!!
00000897 E87D3E                     call    disable_autoexec        ; no, we should disable it
                                process_autoexec:
                                    ; !!!
0000089A E8C53E                     call    CheckQueryOpt  ; MSDOS 6.21 IO.SYS - SYSINIT:0827h
                                    ;mov     cl,[command_line]
                                    ; 30/12/2022
0000089D BE[BB4B]                   mov     si,command_line
000008A0 8A0C                       mov     cl,[si]
000008A2 B500                       mov     ch,0
000008A4 41                         inc     cx
                                    ;mov    si,command_line
000008A5 01CE                       add     si,cx
000008A7 C6040D                     mov     byte [si],cr ; 0Dh    ; cr-terminate command line
                                    ;;;;
                                ; 30/12/2022 - Retro DOS v4.2
                                %if 0
                                    ;mov    si,(offset command_line+1)
                                    mov     si,command_line+1
                                    push    ds
                                    pop     es
                                    mov     di,si
                                    mov     cl,0FFh ; -1
                                _@_loop:
                                    inc     cl ; +1
                                    lodsb
                                    stosb
                                    or      al,al
                                    jnz     short _@_loop
                                    dec     di
                                    mov     al,0Dh
                                    stosb                   ; cr-terminate command line
                                    mov     [command_line],cl ; command line length (except CR)
                                %endif
                                ; ---------------------------------------------------------------------------
                                ;   Once we get to this point, the above code, which is below "retry"
                                ;   in memory, can be trashed (and in fact is -- see references to retry
                                ;   which follow....)
                                retry:              ; PCDOS 7.1 IBMBIO.COM - SYSINIT:094Ch ; 07/04/2024
000008AA BA[2D4B]                   mov     dx,commnd       ; now pointing to file description
                                ; we are going to open the command interpreter and size it as is done in
                                ; ldfil. the reason we must do this is that sysinit is in free memory. if
                                ; there is not enough room for the command interpreter,exec will probably
                                ; overlay our stack and code so when it returns with an error sysinit won't be
                                ; here to catch it. this code is not perfect (for instance .exe command
                                ; interpreters are possible) because it does its sizing based on the
                                ; assumption that the file being loaded is a .com file. it is close enough to
                                ; correctness to be usable.
                                ; first, find out where the command interpreter is going to go.
000008AD 52                         push    dx              ; save pointer to name
000008AE BBFFFF                     mov     bx,0FFFFh
000008B1 B448                       mov     ah,48h  ; ALLOC
000008B3 CD21                           int     21h             ; get biggest piece
000008B5 B448                       mov     ah,48h  ; ALLOC
000008B7 CD21                       int     21h             ; second time gets it
000008B9 726B                       jc      short memerrjx ; oooops
000008BB 8EC0                       mov     es,ax
000008BD B449                       mov     ah,49h  ; DEALLOC
000008BF CD21                       int     21h             ; give it right back
000008C1 89DD                       mov     bp,bx
                                ; es:0 points to block,and bp is the size of the block in para.
                                ; we will now adjust the size in bp down by the size of sysinit.
                                ; we need to do this because exec might get upset if some of the exec
                                ; data in sysinit is overlayed during the exec.
                                    ; 22/10/2022
                                    ; (MSDOS 5.0 IO.SYS SYSINIT:083Bh)
000008C3 8B1E[9402]                     mov     bx,[MEMORY_SIZE] ; get location of end of memory
000008C7 8CC8                       mov     ax,cs           ; get location of beginning of sysinit
                                ; Note that the "config_wrkseg" environment data is a segment in
                                ; unallocated memory (as of the Dealloc of [area], above). This is ideal
                                ; in one sense, because Exec is going to make a copy of it for COMMAND.COM
                                ; anyway, and no one has responsibility for freeing "config_wrkseg". But
                                ; we need to make sure that there's no way Exec will stomp on that data
                                ; before it can copy it, and one way to do that is to make the available
                                ; memory calculation even more "paranoid", by subtracting "config_wrkseg"
                                ; from the "memory_size" segment value (which is typically A000h) instead
                                ; of the current sysinit CS....
                                ;
                                ; The reason I use the term "paranoid" is because this code should have
                                ; slid the data required by Exec up to the very top of memory, because as
                                ; it stands, you have to have sizeof(COMMAND.COM) PLUS 64K to load just
                                ; COMMAND.COM (64k is about what sysinit, and all the goop above sysinit,
                                ; consumes). Now it's just a little worse (65K or more, depending on
                                ; the size of your CONFIG.SYS, since the size of the environment workspace
                                ; is determined by the size of CONFIG.SYS.... -JTP
                                    ; 30/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21, IO.SYS)
                                    ; (SYSINIT:0858h)
000008C9 8B0E[6019]                 mov     cx,[config_envlen]
000008CD E303                           jcxz    no_env          ; use config_wrkseg only if there's env data
```

```
25038 000008CF A1[6219]                       mov ax,[config_wrkseg]
25039
25040                                     ; 22/10/2022
25041                                     ;mov    cx,[config_envlen]
25042                                         ;jcxz      no_env         ; use config_wrkseg only if there's env data
25043                                         ;mov       ax,[config_wrkseg]
25044                                 ;no_env:
25045                                     ; 22/10/2022
25046                                     ; (MSDOS 5.0 IO.SYS SYSINIT:0841h)
25047                                 no_env:
25048                                     ; 30/12/2022
25049                                     ; (MSDOS 6.21 IO.SYS SYSINIT:0861h)
25050 000008D2 29C3                     sub     bx,ax           ; bx is size of sysinit in para
25051 000008D4 83C311                   add     bx,11h          ; add the sysinit php
25052 000008D7 29DD                     sub     bp,bx           ; sub sysinit size from amount of free memory
25053 000008D9 724B                     jc      short memerrjx ; if there isn't even this much memory, give up
25054
25055                                         ;mov       ax,(OPEN<<8)   ; open the file being execed
25056 000008DB B8003D                       mov ax,3D00h
25057 000008DE F9                       stc                     ; in case of int 24
25058 000008DF CD21                     int     21h
25059 000008E1 7271                     jc      short comerr   ; ooops
25060                                                   ; DOS - 2+ - OPEN DISK FILE WITH HANDLE
25061                                                   ; DS:DX -> ASCIZ filename
25062                                                   ; AL = access mode
25063                                                   ; 0 - read
25064                                     ; 22/10/2022
25065                                     ; (MSDOS 5.0 IO.SYS SYSINIT:0852h)
25066 000008E3 89C3                         mov     bx,ax           ; handle in bx
25067
25068                                 ;   If the standard command interpreter is being used, verify it is correct
25069
25070                                     ; 30/12/2022 - Retro DOS v4.2
25071                                     ; (MSDOS 6.21 IO.SYS, SYSINIT:0874h)
25072 000008E5 803E[2A4B]00             cmp     byte [newcmd],0       ; was a new shell selected?
25073 000008EA 7518                     jne     short skip_validation ; yes
25074                                     ; 07/04/2024 - Retro DOS v5.0
25075                                     ; (PCDOS 7.1 IBMBIO.COM, SYSINIT:098Eh)
25076 000008EC BA[A608]                 mov     dx,retry-4     ; PCDOS 7.1 IBMBIO.COM - SYSINIT:0948h
25077 000008EF B90400                   mov     cx,4           ;
25078 000008F2 B43F                     mov     ah,READ        ;
25079 000008F4 CD21                     int     21h            ;
25080 000008F6 803E[A608]E9             cmp     byte [retry-4],0E9h
25081 000008FB 7557                     jne     short comerr
25082                                     ; 20/04/2019 - Retro DOS v4.0
25083                                     ; 30/12/2022
25084                                     ;cmp    byte [retry-1],64h ; MSDOS 6.21 IO.SYS - SYSINIT:088Ch
25085                                                                ; .. COMMAND.COM Version 6.20 (14h&0Fh)
25086                                     ; 07/04/2024 - Retro DOS v5.0
25087                                     ;;cmp   byte [retry-1],66h ; .. COMMAND.COM Version 6.22 (16h&0Fh)
25088                                     ;cmp    byte [retry-1],7Ah ; PCDOS 7.1 IBMBIO.COM - SYSINIT:099Fh
25089                                                                ; .. COMMAND.COM Version 7.10 (0Ah&0Fh)
25090 000008FD 803E[A908]7A             cmp     byte [retry-1],((MAJOR_VERSION&0Fh)<<4)|(MINOR_VERSION&0Fh)
25091 00000902 7550                     jne     short comerr   ;
25092
25093                                     ; 22/10/2022
25094                                     ;cmp    byte [newcmd],0       ; was a new shell selected?
25095                                     ;jne    short skip_validation ; yes
25096                                     ;mov    dx,retry-4
25097                                     ;mov    cx,4           ;
25098                                     ;mov    ah,READ        ;
25099                                     ;int    21h            ;
25100                                     ;cmp    byte [retry-4],0E9h
25101                                     ;jne    short comerr
25102                                     ;; 20/04/2019 - Retro DOS v4.0
25103                                     ;cmp    byte [retry-1],64h ; MSDOS 6.21 IO.SYS - SYSINIT:088Ch
25104                                     ;;cmp   byte [retry-1],((MAJOR_VERSION&0Fh)<<4)|(MINOR_VERSION&0Fh)
25105                                     ;jne    short comerr   ;
25106
25107                                 ;skip_validation:
25108                                     ; 22/10/2022
25109                                     ; (MSDOS 5.0 IO.SYS SYSINIT:0854h)
25110                                 skip_validation:
25111                                     ; 30/12/2022
25112                                     ; (MSDOS 6.21 IO.SYS SYSINIT:0893h)
25113 00000904 31C9                     xor     cx,cx
25114 00000906 31D2                     xor     dx,dx
25115                                     ;mov    ax,(LSEEK<<8)|2
25116 00000908 B80242                   mov     ax,4202h
25117 0000090B F9                       stc                     ; in case of int 24
25118 0000090C CD21                     int     21h             ; get file size in dx:ax
25119 0000090E 7244                     jc      short comerr
25120                                                   ; convert size in dx:ax to para in ax
25121 00000910 83C00F                   add     ax,15           ; round up size for conversion to para
25122 00000913 83D200                   adc     dx,0
25123 00000916 E88104                   call    off_to_para
25124 00000919 B10C                     mov     cl,12
25125 0000091B D3E2                     shl     dx,cl           ; low nibble of dx to high nibble
25126 0000091D 09D0                     or      ax,dx           ; ax is now # of para for file
25127 0000091F 83C010                   add     ax,10h          ; 100h byte php
25128 00000922 39E8                     cmp     ax,bp           ; will command fit in available mem?
25129 00000924 7208                     jb      short okld      ; jump if yes.
25130
25131                                 ; 30/12/2022
25132                                 %if 0
25133                                     ; 22/10/2022
25134                                 memerrjx:   ; (MSDOS 5.0 IO.SYS SYSINIT:0876h)
25135                                     ;jmp    memerr  ; (MSDOS 5.0 IO.SYS SYSINIT:34D5h)
25136                                     ; 02/11/2022
25137                                     ;jmp    mem_err
25138                                     ; 11/12/2022
25139                                     ; ds = cs
25140                                     jmp     mem_err2
25141                                 %endif
25142                                     ; 30/12/2022
25143                                     ; (MSDOS 6.21, IO.SYS, SYSINIT:08B5h)
25144                                 memerrjx:
25145 00000926 BA[4951]                 mov     dx,badmem      ; "Configuration too large for memory"
25146 00000929 E82841                   call    print
25147 0000092C EB3A                     jmp     short continue
25148
25149                                 okld:
25150 0000092E B43E                     mov     ah,3Eh ; CLOSE
25151 00000930 CD21                     int     21h            ; close file
25152
25153                                     ; 22/10/2022
25154 00000932 5A                       pop     dx      ; (MSDOS 5.0 IO.SYS SYSINIT:087Dh)
25155
25156                                     ; 24/03/2019
25157
25158 00000933 0E                        push    cs             ; point es to sysinitseg
25159 00000934 07                       pop     es
25160 00000935 BB[BF02]                     mov     bx,COMEXE ; point to exec block
25161                                     ; 22/10/2022
```

```
                                    ;pop    dx              ; recover pointer to name
                                ;;ifdef     MULTI_CONFIG
                                ;   If there's any environment data in "config_wrkseg", pass it to shell;
                                ;   there will be data if there were any valid SET commands and/or if a menu
                                ;   selection was made (in which case the CONFIG environment variable will be
                                ;   set to that selection).
                                    ; 23/10/2022
                                    ;mov    cx,[config_envlen]
                                    ;jcxz   no_envdata
                                        ;mov       cx,[config_wrkseg]
                                ;no_envdata:
                                    ;;mov   [bx+EXEC0.ENVIRON],cx
                                    ;mov    [bx],cx
                                ;;endif      ;MULTI_CONFIG
                                    ; 30/12/2022 - Retro DOS v4.2
                                    ; (MSDOS 6.21 IO.SYS SYSINIT:08C7h)
00000938 8B0E[6019]                 mov     cx,[config_envlen]
0000093C E304                       jcxz    no_envdata
0000093E 8B0E[6219]                     mov cx,[config_wrkseg]
                                no_envdata:
                                    ;mov    [bx+EXEC0.ENVIRON],cx
00000942 890F                       mov     [bx],cx
                                    ; 23/10/2022
                                    ; (MSDOS 5.0 IO.SYS SYSINIT:0883h)
                                    ;mov    [bx+EXEC0.COM_LINE+2],cs ; set segments
00000944 8C4F04                     mov     [bx+4],cs
                                    ;mov    [bx+EXEC0.5C_FCB+2],cs
00000947 8C4F08                     mov     [bx+8],cs
                                    ;mov    [bx+EXEC0.6C_FCB+2],cs
0000094A 8C4F0C                     mov     [bx+12],cs
                                    ;mov    ax,(EXEC<<8) + 0
                                    ; 23/10/2022
                                    ;xor    ax,ax
                                    ;mov    ah,4Bh
                                    ; 04/07/2023
                                    ;mov    ax,4B00h
0000094D B8004B                     mov     ax,(EXEC<<8)
00000950 F9                         stc                     ; in case of int 24
00000951 CD21                           int     21h             ; go start up command
                                                            ; DOS - 2+ - LOAD OR EXECUTE (EXEC)
                                                            ; DS:DX -> ASCIZ filename
                                                            ; ES:BX -> parameter block
                                                            ; AL = subfunc: load & execute program
                                    ;push   cs
                                    ;pop    ds
                                    ; 13/04/2024
                                    ; 23/10/2022
00000953 52                         push    dx              ; push to balance fall-through pop
                                ; note fall through if exec returns (an error)
                                comerr:
                                    ; 23/10/2022
                                ;;ifdef     MULTI_CONFIG
                                    ;cmp    byte [commnd4],0
                                    ;je     short comerr2  ; all defaults exhausted, print err msg
                                    ;cmp    byte [newcmd],0
                                    ;je     short continue ; don't print err msg for defaults just yet
                                ;comerr2:
                                ;;endif
                                    ; 30/12/2022 - Retro DOS v4.2
                                    ;push   cs
                                    ;pop    ds
                                    ; 07/04/2024
                                    ; ds = cs
00000954 803E[9E4B]00               cmp     byte [commnd4],0
00000959 7407                       je      short comerr2  ; all defaults exhausted, print err msg
0000095B 803E[2A4B]00               cmp     byte [newcmd],0
00000960 7406                       je      short continue ; don't print err msg for defaults just yet
                                comerr2:
                                    ; 07/04/2024
                                    ;push   dx ; 30/12/2022
                                    ; 23/10/2022
00000962 BA[C550]                       mov     dx,badcom ; want to print command error
00000965 E8C040                     call    badfil
                                    ; 07/04/2024
                                    ;pop    dx  ; 30/12/2022
                                continue:
                                    ; 13/04/2024
                                    ; 23/10/2022
00000968 5A                         pop     dx
                                ; 30/12/2022
                                %if 0
                                ;;ifndef MULTI_CONFIG
                                    ;jmp    stall
                                    ; 24/10/2022
                                stall:              ; (MSDOS 5.0 IO.SYS, SYSINIT:0899h)
                                    jmp     short stall
                                ;;else
                                %endif
                                ; 30/12/2022 (MSDOS 6.21 SYSINIT, Retro DOS v4.2)
                                ;%if 1
                                ; 23/10/2022 (MSDOS 5.0 SYSINIT, Retrodos v4.0)
                                ;%if 0
00000969 B419                       mov     ah,GET_DEFAULT_DRIVE ; 19h
0000096B CD21                       int     21h             ;
0000096D 0441                       add     al,'A'          ;
0000096F 88C2                       mov     dl,al           ; DL == default drive letter
00000971 BE[6D4B]                   mov     si,commnd2
00000974 803E[2A4B]00               cmp     byte [newcmd],0 ; if a SHELL= was given
00000979 7505                       jne     short do_def2  ; then try the 2nd alternate;
0000097B C60400                     mov     byte [si],0    ; otherwise, the default SHELL= was tried,
0000097E EB05                       jmp     short do_def3   ; which is the same as our 2nd alt, so skip it
                                do_def2:
00000980 803C00                     cmp     byte [si],0    ; has 2nd alternate been tried?
00000983 7554                           jne short do_alt    ; no
                                do_def3:
```

```
                            mov     si,commnd3
                            cmp     byte [si],0     ; has 3rd alternate been tried?
                            jne     short do_alt    ; no
                            mov     si,commnd4
                            cmp     byte [si],0     ; has 4th alternate been tried?
                            jne     short do_alt    ; no
                            push    dx              ;
                            mov     dx,badcomprmpt
                            call    print           ;
                            pop     dx              ; recover default drive letter in DL
                        request_input:              ;
                            mov     ah,STD_CON_OUTPUT
                            int     21h             ;
                            push    dx              ;
                            mov     dl,'>'          ;
                            int     21h             ;
                            mov     bl,[tmplate+1] ; [tmplate+1] = 12
                            mov     bh,0            ;
                            mov     byte [commnd+bx],0Dh
                            mov     dx,tmplate
                            mov     ah,STD_CON_STRING_INPUT
                            int     21h             ; read a line of input
                            mov     dx,crlfm        ;
                            call    print           ;
                            pop     dx              ;
                            mov     bl,[tmplate+1] ;
                            or      bl,bl           ; was anything typed?
                            jz      short request_input ;
                            mov     byte [newcmd],1 ; disable validation for user-specified binaries
                            mov     byte [commnd+bx],0 ; NULL-terminate it before execing it
                            mov     word [command_line],0D00h
                            jmp     short do_exec   ;
                        do_alt:
                            push    ds
                            pop     es
                            mov     byte [newcmd],0 ; force validation for alternate binaries
                            mov     di,commnd       ;
                        do_alt1:
                            lodsb                   ; copy the alternate, zapping it as we go,
                            mov     byte [si-1],0   ; so that we know it's been tried
                            stosb                   ;
                            or      al,al           ;
                            jnz     short do_alt1   ;
                            mov     di,command_line
                            cmp     byte [si+2],':'
                            jne     short do_alt2   ;
                            mov     [si+1],dl       ; stuff default drive into alt. command line
                        do_alt2:                    ;
                            lodsb                   ;
                            stosb                   ;
                            or      al,al           ;
                            jnz     short do_alt2   ;
                            mov     byte [di-1],cr

                        ;;   Last but not least, see if we need to call disable_autoexec

                            ; MSDOS 6.0 (SYSINIT1.ASM)
                            ;cmp    [command_line-1],0
                                ;jne       short do_exec   ;
                                ;mov       [command_line-1],'/'
                            ;call   disable_autoexec ;

                            ; MSDOS 6.21 IO.SYS (SYSINIT:0994h)
00000A03 C606[7B4C]00       mov     byte [dae_flag],0 ; 24/03/2019 - Retro DOS v4.0
00000A08 E80C3D             call    disable_autoexec
00000A0B E8543D             call    CheckQueryOpt  ; 24/03/2019 - Retro DOS v4.0
                        do_exec:
00000A0E E999FE             jmp     retry           ;

                        ;;endif     ;MULTI_CONFIG

                        ;%endif ; 23/10/2022 (MSDOS 5.0 SYSINIT)
                        ;%endif ; 30/12/2022 (MSDOS 6.21 SYSINIT)

                        ; 24/03/2019 - Retro DOS v4.0

                        ; ----------------------------------------------------------------------
                        ; procedure : AllocFreeMem
                        ;
                        ; Allocate Max memory from DOS to find out where to load DOS.
                        ; DOS is at temporary location when this call is being made
                        ;
                        ; Inputs : None
                        ; Outputs: The biggest chunk of memory is allocated (all mem at init time)
                        ;       [area] & [memhi] set to the para value of the start of the
                        ;       free memory.
                        ;
                        ; Uses   : AX, BX
                        ;
                        ; ----------------------------------------------------------------------

                            ; 30/12/2022 - Retro DOS v4.2
                            ; (MSDOS 6.21 IO.SYS, SYSINIT:09A2h)

                            ; 08/04/2024 - Retro DOS v5.0
                            ; (PCDOS 7.1 IBMBIO.COM, SYSINIT:0AB5h)

                            ; 23/10/2022
                        AllocFreeMem:
00000A11 BBFFFF             mov     bx,0FFFFh
00000A14 B448               mov     ah,48h ; ALLOC
00000A16 CD21               int     21h                     ; first time fails
00000A18 B448               mov     ah,48h ; ALLOC
00000A1A CD21               int     21h                     ; second time gets it
                            ; 11/12/2022
                            ; ds = cs
                            ;mov    [cs:area],ax
                            ;mov    [cs:memhi],ax           ; memhi:memlo now points to
00000A1C A3[6803]           mov     [area],ax
00000A1F A3[6403]           mov     [memhi],ax              ; memhi:memlo now points to
00000A22 C3                 retn                            ; start of free memory

                            ; include msbio.cl6
                        ; ----------------------------------------------------------------------
                        DOSLOMSG:
00000A23 484D41206E6F742061-    db      'HMA not available: Loading DOS low',0Dh,0Ah,'$'
00000A2C 7661696C61626C653A-
00000A35 204C6F6164696E6720-
00000A3E 444F53206C6F770D0A-
00000A47 24

                        FEmsg:
00000A48 466174616C20457272-    db      'Fatal Error: Cannot allocate Memory for DOS',0Dh,0Ah,'$'
00000A51 6F723A2043616E6E6F-
00000A5A 7420616C6C6F636174-
```

```
25403 00000A63 65204D656D6F727920-
25403 00000A6C 666F7220444F530D0A-
25403 00000A75 24
25404
25405                                 ; --------------------------------------------------------------------
25406                                 ;
25407                                 ; procedure : LoadDOSHiOrLo
25408                                 ;
25409                                 ;         Tries to move DOS into HMA. If it fails then loads
25410                                 ;         DOS into Low memory. For ROMDOS, nothing is actually
25411                                 ;         moved; this just tries to allocate the HMA, and prints
25412                                 ;         a message if this is not possible.
25413                                 ;
25414                                 ; --------------------------------------------------------------------
25415
25416                                     ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
25417                                 LoadDOSHiOrLo:
25418                                     ; 27/03/2019 - Retro DOS v4.0
25419                                     ; ds = cs
25420 00000A76 E81F00                   call    TryToMovDOSHi         ; Try moving it into HMA (M024)
25421                                     ;jc     short LdngLo          ; If that don't work...
25422                                     ;retn
25423                                     ; 18/12/2022
25424 00000A79 731C                     jnc     short LoadDosHi_ok
25425                                 LdngLo:
25426                                     ; 23/10/2022
25427                                     ;push   cs
25428                                     ;pop    ds
25429                                     ; 11/12/2022
25430                                     ; ds = cs
25431 00000A7B B409                     mov     ah,9
25432 00000A7D BA[230A]                 mov     dx,DOSLOMSG           ; inform user that we are
25433 00000A80 CD21                     int     21h                   ; loading low
25434
25435                                 ;ifndef ROMDOS
25436                                     ; actually move the dos, and reinitialize it.
25437
25438 00000A82 BB0100                   mov     bx,1                          ; M012
25439                                                                   ;  use int 21 alloc for mem
25440 00000A85 E83F00                   call    MovDOSLo
25441                                     ; 11/12/2022
25442                                     ; ds = cs
25443                                     ;mov    es,[cs:CURRENT_DOS_LOCATION]  ; give dos its temporary loc.
25444                                     ; 23/10/2022
25445 00000A88 8E06[7302]               mov     es,[CURRENT_DOS_LOCATION]
25446                                     ;;mov   es,[cs:FINAL_DOS_LOCATION]  ; 24/03/2019 - Retro DOS v4.0
25447                                     ;mov    es,[FINAL_DOS_LOCATION] ; 27/03/2019
25448 00000A8C 31C0                     xor     ax,ax                         ; ax = 00 ---> install stub
25449                                     ; 11/12/2022
25450                                     ; ds = cs
25451                                     ;call   far [cs:dos_segreinit]        ; call dos segreinit
25452 00000A8E FF1E[7D02]               call    far [dos_segreinit] ; 27/03/2019
25453
25454                                 ;endif ; ROMDOS
25455                                     ; 23/10/2022
25456                                     ;mov    byte [cs:runhigh],0           ; mark that we are running lo
25457                                     ; 11/12/2022
25458                                     ; ds = cs
25459 00000A92 C606[6C02]00             mov     byte [runhigh],0 ; 27/03/2019
25460                                 LoadDosHi_ok:       ; 18/12/2022
25461 00000A97 C3                       retn
25462
25463                                 ; --------------------------------------------------------------------
25464                                 ;
25465                                 ; procedure : TryToMovDOSHi
25466                                 ;
25467                                 ;         This tries to move DOS into HMA.
25468                                 ;         Returns CY if it failed.
25469                                 ;         If it succeeds returns with carry cleared.
25470                                 ;
25471                                 ;         For ROMDOS, dos_segreinit must be called again to allow
25472                                 ;         the A20 switching code in the low mem stub to be installed.
25473                                 ;
25474                                 ; --------------------------------------------------------------------
25475
25476                                     ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
25477                                     ; (MSDOS 5.0 IO.SYS - SYSINIT:092Ah)
25478                                 TryToMovDOSHi:
25479                                     ; 11/12/2022
25480                                     ; 27/03/2019 - Retro DOS v4.0
25481                                     ; ds = cs
25482 00000A98 E81300                   call    MovDOSHi
25483 00000A9B 7210                     jc      short ttldhx
25484
25485                                 ;ifndef ROMDOS
25486                                     ; 23/10/2022
25487                                     ;mov    es,[cs:CURRENT_DOS_LOCATION] ; give dos its temporary loc.
25488                                     ;;mov   es,[cs:FINAL_DOS_LOCATION] ; 24/03/2019 - Retro DOS v4.0
25489                                     ; 11/12/2022
25490                                     ; ds = cs
25491 00000A9D 8E06[7302]               mov     es,[CURRENT_DOS_LOCATION]
25492                                 ;else
25493                                 ;       ..
25494                                 ;endif ; ROMDOS
25495
25496                                     ; 11/12/2022
25497                                     ; ds = cs
25498 00000AA1 31C0                     xor     ax,ax                 ; ax = 00 ---> install stub
25499                                     ;call   far [cs:dos_segreinit]; call dos segreinit
25500 00000AA3 FF1E[7D02]               call    far [dos_segreinit]
25501                                     ;mov    byte [cs:runhigh],1
25502 00000AA7 C606[6C02]01             mov     byte [runhigh],1
25503 00000AAC F8                       clc
25504                                 ttldhx:
25505 00000AAD C3                       retn
25506
25507                                 ; --------------------------------------------------------------------
25508                                 ;
25509                                 ; procedure : MovDOSHi
25510                                 ;
25511                                 ;         Tries to allocate HMA and Move DOS/BIOS code into HMA
25512                                 ;         For ROMDOS, the code is not actually moved, but the
25513                                 ;         HMA is allocated and prepared for sub-allocation.
25514                                 ;
25515                                 ;         Returns : CY if it failed
25516                                 ;
25517                                 ; --------------------------------------------------------------------
25518
25519                                     ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
25520                                 MovDOSHi:
25521                                     ; 14/05/2019
25522                                     ; 27/03/2019 - Retro DOS v4.0
25523                                     ; ds = cs
25524 00000AAE E8D600                   call    AllocHMA
```

```
00000AB1 7213                       jc      short mdhx                  ; did we get HMA?
00000AB3 B8FFFF                     mov     ax,0FFFFh                   ; yes, HMA seg = 0ffffh
00000AB6 8EC0                       mov     es,ax

                                ;ifndef ROMDOS
                                    ; actually move the BIOS and DOS

                                    ; NOTE: Retro DOS v4.0 does not move BIOS (IO.SYS) to HMA
                                    ; 24/03/2019

                                    ; 23/10/2022
00000AB8 E83200                     call    MovBIOS                     ; First move BIOS into HMA

                                    ; ES:DI points to free HMA after BIOS

                                    ; 14/05/2019
                                    ; 24/03/2019 - Retro DOS v4.0
                                    ;xor    di,di

                                    ; 23/10/2022
                                    ;mov    cx,[cs:hi_doscod_size]      ; pass the code size of DOS
                                    ; 11/12/2022
                                    ; ds = cs
00000ABB 8B0E[8302]                 mov     cx,[hi_doscod_size]         ; when it is in HMA
00000ABF E81100                     call    MovDOS                      ; and move it

                                    ; ES:DI points to free HMA after DOS
                                ;else
                                ;   ; allocate space at beginning of HMA to allow for CPMHack
                                ;
                                ;   mov     di,0E0h                     ; room for 5 bytes at ffff:d0
                                ;
                                ;endif ; ROMDOS

00000AC2 E87602                     call    SaveFreeHMAPtr              ; Save the Free HMA ptr
00000AC5 F8                         clc
                                mdhx:
00000AC6 C3                         retn

                                ; ---------------------------------------------------------------------------
                                ;
                                ; procedure : MovDOSLo
                                ;
                                ;           Allocates memory from DOS and moves BIOS/DOS code into it
                                ;
                                ; ---------------------------------------------------------------------------

                                    ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)

                                ;ifndef ROMDOS

                                MovDOSLo:
                                    ; 14/05/2019
                                    ; 27/03/2019 - Retro DOS v4.0
                                    ; ds = cs
00000AC7 E84500                     call    AllocMemForDOS              ; incestuosly!!!

                                    ; 23/10/2022
                                    ; 14/05/2019
                                    ;inc    ax  ; skip MCB

00000ACA 8EC0                       mov     es,ax                       ; pass the segment to MovBIOS
                                    ; NOTE: Retro DOS v4.0 does not move BIOS (IO.SYS) to HMA
                                    ; 24/03/2019

                                    ; 23/10/2022
00000ACC E81E00                     call    MovBIOS

                                ;------ ES:DI points memory immediately after BIOS

                                    ; 14/05/2019
                                    ; NOTE:
                                    ;     Order of (RETRO) DOS kernel sections at memory:
                                    ;       BIOSDATA+BIOSCODE+BIOSDATAINIT+DOSDATA+DOSCODE(LOW)

                                    ; 24/03/2019 - Retro DOS v4.0
                                    ;xor    di,di

                                    ; 23/10/2022
                                    ;mov    cx,[cs:lo_doscod_size]      ; DOS code size when loaded
                                    ; 11/12/2022
                                    ; ds = cs
00000ACF 8B0E[8102]                 mov     cx,[lo_doscod_size]         ; low
                                    ;call   MovDOS
                                    ;retn
                                    ; 11/12/2022
                                    ;jmp    short MovDOS

                                ;endif ; ROMDOS

                                ; 11/12/2022

                                ; ---------------------------------------------------------------------------
                                ;
                                ; procedure : MovDOS
                                ;
                                ;           Moves DOS code into requested area
                                ;
                                ;   In : ES:DI - pointer to memory where DOS is to be moved
                                ;        CX    - size of DOS code to be moved
                                ;
                                ;   Out : ES:DI - pointer to memory immediately after DOS
                                ;
                                ; ---------------------------------------------------------------------------

                                    ; 11/12/2022
                                    ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)

                                ;ifndef ROMDOS

                                MovDOS:
                                    ; 14/05/2019
                                    ; 27/03/2019 - Retro DOS v4.0

                                    ; 11/12/2022
                                    ; ds = cs

                                    ; 23/10/2022
                                    ;push   ds ; *//

00000AD3 06                         push    es
00000AD4 57                         push    di

                                    ; 11/12/2022
```

```
00000AD5 1E                         push    ds ; *// ; 11/12/202
                                    ; 29/04/2019
00000AD6 C536[7102]                 lds     si,[dosinit] ; 11/12/2022
                                    ; 23/10/2022
                                    ;lds    si,[cs:dosinit]
                                    ; 03/09/2023
00000ADA 89F0                       mov     ax,si
00000ADC F3A4                       rep     movsb
00000ADE 1F                         pop     ds ; *// ; 11/12/2022
00000ADF 5B                         pop     bx                          ; get back offset into which
                                                                        ;  DOS was moved
                                    ; 03/09/2023
                                    ;;mov   ax,[cs:dosinit]                     ; get the offset at which DOS
                                                                        ;  wants to run
                                    ; 03/09/2023
                                    ;mov    ax,[dosinit]
                                    ; ax = [dosinit]
00000AE0 29D8                       sub     ax,bx
00000AE2 E8B502                     call    off_to_para
00000AE5 5B                         pop     bx                          ; get the segment at which
                                                                        ;  we moved DOS into
00000AE6 29C3                       sub     bx,ax                       ; Adjust segment
                                    ; 11/12/2022
                                    ; 23/10/2022
                                    ;mov    [cs:CURRENT_DOS_LOCATION],bx ; and save it
                                    ;;mov   [cs:FINAL_DOS_LOCATION],bx
                                    ; 11/12/2022
00000AE8 891E[7302]                 mov     [CURRENT_DOS_LOCATION],bx
                                    ; 27/03/2019
                                    ;pop    ds ; *//
                                    ; ds = cs
                                        ;mov    [FINAL_DOS_LOCATION],bx
00000AEC C3                             retn
                                ;endif ;ROMDOS
                                ; NOTE: Retro DOS v4.0 does not move BIOS (IO.SYS) to HMA
                                ; 24/03/2019
                                ; ---------------------------------------------------------------------------
                                ;
                                ; procedure : MovBIOS
                                ;
                                ;           Moves BIOS code into requested segment
                                ;
                                ;   In : ES - segment to which BIOS is to be moved
                                ;             ( it moves always into offset BCode_Start)
                                ;
                                ;   Out : ES:DI - pointer to memory immediately after BIOS
                                ;
                                ; ---------------------------------------------------------------------------
                                    ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; 08/04/2024 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
                                ;ifndef ROMDOS
                                MovBIOS: ; proc    near
                                    ; 11/12/2022
00000AED 1E                         push    ds ; ds = cs
                                    ;
                                    ; 23/10/2022
                                    ;mov    ds,[cs:temp_bcode_seg]      ; current BIOS code seg
                                    ; 17/09/2023 ; 08/04/2024
00000AEE 8E1E[8902]                 mov     ds,[temp_bcode_seg]
                                    ;mov    si,BCODE_START ; mov si,30h
                                    ; 09/12/2022
00000AF2 BE[3000]                   mov     si,BCODESTART ; 30h
00000AF5 89F7                       mov     di,si
                                    ;mov    cx,1E00h ; PCDOS 7.1 IBMBIO.COM ; 08/04/2024
00000AF7 B9701D                     mov     cx,BCODE_END ; mov cx,1A60h
00000AFA 29F1                       sub     cx,si                       ; size of BIOS
00000AFC D1E9                       shr     cx,1                        ; Both the labels are para
                                                                        ;  aligned
00000AFE F3A5                       rep     movsw
                                    ; 11/12/2022
00000B00 1F                         pop     ds ; ds = cs
                                    ;
00000B01 06                         push    es
00000B02 57                         push    di                          ; save end of BIOS
00000B03 8CC0                       mov     ax,es
                                    ;
                                    ; 11/12/2022
                                    ;mov    [cs:BCodeSeg],ax            ; save it for later use
                                    ;;call  dword ptr cs:_seg_reinit_ptr
                                    ;call   far [cs:seg_reinit_ptr]             ; far call to seg_reinit (M022)
                                    ; ds = cs
00000B05 A3[8A03]                   mov     [BCodeSeg],ax
00000B08 FF1E[8702]                 call    far [seg_reinit_ptr]
                                    ;
00000B0C 5F                         pop     di
00000B0D 07                         pop     es                          ; get back end of BIOS
00000B0E C3                         retn
                                ;MovBIOS endp
                                ;endif ; ROMDOS
                                ; 11/12/2022
                                %if 0
                                ; 24/03/2019
                                ; ---------------------------------------------------------------------------
                                ;
                                ; procedure : MovDOS
                                ;
                                ;           Moves DOS code into requested area
                                ;
                                ;   In : ES:DI - pointer to memory where DOS is to be moved
                                ;        CX    - size of DOS code to be moved
                                ;
                                ;   Out : ES:DI - pointer to memory immediately after DOS
                                ;
                                ; ---------------------------------------------------------------------------
```

```
                            ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)

                           ;ifndef ROMDOS

                           MovDOS:
                            ; 14/05/2019
                            ; 27/03/2019 - Retro DOS v4.0

                            ; 11/12/2022
                            ; ds = cs

                            ; 23/10/2022
                            ;push  ds ; *//

                            push   es
                            push   di

                            ; 11/12/2022
                            push   ds ; *// ; 11/12/202

                            ; 29/04/2019
                            lds    si,[dosinit] ; 11/12/2022
                            ; 23/10/2022
                            ;lds   si,[cs:dosinit]
                            ; 03/09/2023
                            mov    ax,si

                            rep    movsb

                            pop    ds ; *// ; 11/12/2022

                            pop    bx                              ; get back offset into which
                                                                   ;  DOS was moved
                            ;mov   ax,[dosinit] ; 03/09/2023
                            ;;mov  ax,[cs:dosinit]                       ; get the offset at which DOS
                                                                   ;  wants to run
                            sub    ax,bx
                            call   off_to_para
                            pop    bx                              ; get the segment at which
                                                                   ;  we moved DOS into
                            sub    bx,ax                           ; Adjust segment

                            ; 11/12/2022
                            ; 23/10/2022
                            ;mov   [cs:CURRENT_DOS_LOCATION],bx  ; and save it
                            ;;mov  [cs:FINAL_DOS_LOCATION],bx
                            ; 11/12/2022
                            mov    [CURRENT_DOS_LOCATION],bx

                            ; 27/03/2019
                            ;pop   ds ; *//
                            ; ds = cs
                            ;mov   [FINAL_DOS_LOCATION],bx

                            retn

                           ;endif ;ROMDOS

                           %endif

                           ; ----------------------------------------------------------------------
                           ;
                           ; procedure : AllocMemForDOS
                           ;
                           ;          Allocate memory for DOS/BIOS code from DOS !!!
                           ;
                           ;   Out : AX - seg of allocated memoryblock
                           ;
                           ; ----------------------------------------------------------------------

                            ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                            ; 08/04/2024 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)

                           ;ifndef ROMDOS

                           AllocMemForDOS:
                            ; 11/12/2022
                            ; 14/05/2019
                            ; 27/03/2019 - Retro DOS v4.0
                            ; ds = cs
                            ;mov   ax,BCode_end
                            ;sub   ax,BCode_start        ; BIOS code size
                            ; 23/10/2022
00000B0F B8701D             mov    ax,BCODE_END ; 1A60h  ; 1A70h for MSDOS 6.21
                                                          ; 30/12/2022
                            ;mov   ax,1E00h ; PCDOS 7.1 IBMBIO.COM ; 08/04/2024
                            ;sub   ax,BCODE_START ; 30h
                            ; 09/12/2022
00000B12 2D[3000]           sub    ax,BCODESTART ; sub ax,30h ; 08/04/2024
                            ; 24/03/2019 - Retro DOS v4.0
                            ; 02/11/2022
                            ;add   ax,[cs:lo_doscod_size]; DOS code size
                            ; 11/12/2022
                            ; ds = cs
00000B15 0306[8102]         add    ax,[lo_doscod_size]
00000B19 83C00F             add    ax,15
00000B1C E87B02             call   off_to_para                  ; convert to para
                            ; 23/10/2022
                            ; 14/05/2019
                            ;inc   ax ; + 1 paragraph for MCB
00000B1F 09DB               or     bx,bx                        ; M012
00000B21 89C3               mov    bx,ax                        ;  can we use int 21 for alloc
00000B23 741A               jz     short update_arena           ; M012
00000B25 B448               mov    ah,48h                       ; request DOS
00000B27 CD21               int    21h
00000B29 7250               jc     short FatalErr               ; IF ERR WE ARE HOSED
                            ; 23/10/2022
                            ; 24/03/2019 - Retro DOS v4.0 (ORG 0)
00000B2B 83E803             sub    ax,3                         ; Take care ORG 30h of
                                                                ;  BIOS code
00000B2E 8EC0               mov    es,ax
                            ;mov   word [es:20h+ARENA.OWNER],08h ; mark it as system
                            ;mov   word [es:20h+ARENA.NAME],'SC' ;  code area
                            ; 14/05/2019
                            ;mov   word [es:ARENA.OWNER],08h     ; mark it as system
                            ;mov   word [es:ARENA.NAME],'SC'     ;  code area
                            ; 08/04/2024 (PCDOS 7.1 IBMBIO.COM)
                            ; 23/10/2022
00000B30 26C70621000800     mov    word [es:20h+1],08h          ; mark it as system
00000B37 26C70628005343     mov    word [es:20h+8],'SC' ; 4353h ;  code area

00000B3E C3                 retn

                           ; BUGBUG -- 5 Aug 92 -- chuckst -- Allocating space for DOS
```

```
                                ;     using DOS itself causes an arena to be generated.
                                ;     Unfortunately, certain programs (like PROTMAN$)
                                ;     assume that the device drivers are loaded into
                                ;     the first arena. For this reason, MagicDrv's
                                ;     main device driver header arena is manually
                                ;     truncated from the arena chain, and the space
                                ;     for DOS is allocated using the following
                                ;     simple code, which also assumes that the
                                ;     first arena is the free one where DOS's low
                                ;     stub will go.
                                ;
                                ; M012 : BEGIN

                                    ; 23/10/2022
                               update_arena:
00000B3F 1E                         push    ds ; ds = cs
00000B40 57                         push    di
00000B41 51                         push    cx
00000B42 52                         push    dx
                                    ; 23/10/2022
                                    ;lds    di,[cs:DOSINFO]                     ; get ptr to DOS var
                                    ; 11/12/2022
                                    ; ds = cs
00000B43 C53E[6D02]                 lds     di,[DOSINFO] ; 27/03/2019
00000B47 4F                         dec     di
00000B48 4F                         dec     di                              ; Arena head is immediately
                                                                            ;  before sysvar
00000B49 8E05                       mov     es,[di]                         ; es = arena head
                                    ;mov    cx,[es:ARENA.SIZE]              ; cx = total low mem size
00000B4B 268B0E0300                 mov     cx,[es:3]
00000B50 39D9                       cmp     cx,bx                           ; is it sufficient ?
00000B52 7227                       jb      short FatalErr                  ; no, fatal error

                                    ;mov    dl,[es:ARENA.SIGNATURE]
00000B54 268A160000                 mov     dl,[es:0]
00000B59 8CC0                       mov     ax,es
00000B5B 01D8                       add     ax,bx                           ; ax = new arena head
00000B5D 8905                       mov     [di],ax                         ; store it in DOS data area
00000B5F 8ED8                       mov     ds,ax
                                    ;mov    [ARENA.SIGNATURE],dl            ; type of arena
00000B61 88160000                   mov     [0],dl
                                    ;mov    word [ARENA.OWNER],0            ; free
00000B65 C70601000000               mov     word [1],0
00000B6B 29D9                       sub     cx,bx                           ; size of the new block
                                    ;mov    [ARENA.SIZE],cx                         ; store it in the arena
00000B6D 890E0300                   mov     [3],cx
00000B71 8CC0                       mov     ax,es                           ; return seg to the caller
                                    ; 23/10/2022
                                    ;; 24/03/2019 - Retro DOS v4.0 (ORG 0)       ; Take care ORG 30h of
00000B73 83E803                     sub     ax,3                            ;  BIOS code
00000B76 5A                         pop     dx
00000B77 59                         pop     cx
00000B78 5F                         pop     di
00000B79 1F                         pop     ds ; ds = cs
00000B7A C3                         retn
                               ;
                               ; M012 : END
                               ;
                               FatalErr:
00000B7B 0E                         push    cs
00000B7C 1F                         pop     ds
00000B7D BA[480A]                   mov     dx,FEmsg
00000B80 B409                       mov     ah,9
00000B82 CD21                       int     21h             ; DOS - PRINT STRING
                                                            ; DS:DX -> string terminated by "$"
                                    ; 30/12/2022 (MSDOS 6.21 SYSINIT)
00000B84 E9C707                     jmp     stall
                                    ; 23/10/2022 (MSDOS 5.0 SYSINIT)
                                    ;cli
                                    ;hlt

                               ;endif ;ROMDOS

                               ; 25/03/2019 - Retro DOS v4.0

                               ; ----------------------------------------------------------------------
                               ;
                               ; procedure : AllocHMA
                               ;
                               ;   grab_the_hma tries to enable a20 and make sure there is memory
                               ;     up there. If it gets any sort of error, it will return with
                               ;     carry set so that we can resort to running low.
                               ;
                               ;   It also returns ES: -> 0ffffh if it returns success
                               ;
                               ; ----------------------------------------------------------------------

                               AllocHMA:
                               ;   cas note:  The pre-286 check is no longer needed here since the
                               ;              presence of XMS is sufficient. However, this code hasn't
                               ;              been deleted because it can be recycled for skipping the
                               ;              extra pass of CONFIG.SYS and assuming we're running low
                               ;              in the case of a pre-286.

                               ;;  see if we're running on a pre-286. If not, force low.
                               ;
                               ;   xor     ax,ax
                               ;   pushf                   ; save flags (like int)
                               ;   push    ax
                               ;   popf
                               ;   pushf
                               ;   pop     ax
                               ;   popf                    ; restore original flags (like int)
                               ;   and     ax,0F000h
                               ;   cmp     ax,0F000h       ; 8088/8086?
                               ;   jz      short grab_hma_error

                                    ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:0A26h)

                                    ; 13/04/2024 - Retro DOS v5.0
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:0C41h)

00000B87 1E                         push    ds
                                    ;;mov   ax,Bios_Data
                                    ;mov    ax,KERNEL_SEGMENT
                                    ; 21/10/2022
00000B88 B87000                     mov     ax,DOSBIODATASEG ; 70h
00000B8B 8ED8                       mov     ds,ax

00000B8D E84A00                     call    IsXMSLoaded
00000B90 7545                       jnz     short grabhma_error

00000B92 B81043                     mov     ax,4310h
```

```
00000B95 CD2F                           int     2Fh             ; get the vector into es:bx
                                                                ; - Multiplex - XMS - GET DRIVER ADDRESS
                                                                ; Return: ES:BX -> driver entry point

00000B97 891E[0E00]                     mov     [xms],bx
                                        ;mov    [0Eh], bx
00000B9B 8C06[1000]                     mov     [xms+2],es
                                        ;mov    [10h],es

00000B9F B401                           mov     ah,1            ; request HMA
00000BA1 BAFFFF                         mov     dx,0FFFFh
                                        ;call   dword ptr ds:0Eh
00000BA4 FF1E[0E00]                     call    far [xms]
00000BA8 48                             dec     ax
00000BA9 7409                           jz      short allocHMA_1 ; error if not able to allocate HMA

                                ;------ Himem may be lying because it has allocated mem for int 15

00000BAB B488                           mov     ah,88h
00000BAD CD15                           int     15h
                                                ; Get Extended Memory Size
                                                ; Return: CF clear on success
                                                ; AX = size of memory above 1M in K
00000BAF 83F840                         cmp     ax,64           ; less than 64 K of hma ?
                                        ;jb     short grabhma_error
                                        ; 11/12/2022
00000BB2 7224                           jb      short grabhma_err ; cf=1
                                allocHMA_1:
00000BB4 B405                           mov     ah,5            ; localenableA20
                                        ;call   dword ptr ds:0Eh
00000BB6 FF1E[0E00]                     call    far [xms]
00000BBA 48                             dec     ax
00000BBB 751A                           jnz     short grabhma_error ; error if couldn't enable A20

00000BBD E89D01                         call    IsVDiskInstalled
00000BC0 7415                           jz      short grabhma_error ; yes, we cant use HMA

00000BC2 B8FFFF                         mov     ax,0FFFFh
00000BC5 8EC0                           mov     es,ax
00000BC7 26C70610003412                 mov     word [es:10h],1234h ; see if we can really read/write there
00000BCE 26813E10003412                 cmp     word [es:10h],1234h
                                        ;jne    short grabhma_error ; don't try to load there if XMS lied
                                        ; 11/12/2022
00000BD5 7401                           je      short allocHMA_ok

                                ; 11/12/2022
                                ;   ; 11/12/2022
                                ;   ; cf=0
                                ;   ;clc
                                ;   pop     ds
                                ;   retn

                                grabhma_error:
00000BD7 F9                             stc
                                        ; 11/12/022
                                grabhma_err:        ; cf=1
                                allocHMA_ok:        ; cf=0
00000BD8 1F                             pop     ds
00000BD9 C3                             retn

                                ; ---------------------------------------------------------------------------
                                ;
                                ; procedure : IsXMSLoaded
                                ;
                                ;             Checks whether a XMS driver is loaded
                                ;
                                ; Returns : Z flag set if XMS driver loaded
                                ;           Z flag reset if no XMS drivers are present
                                ;
                                ; ---------------------------------------------------------------------------

                                IsXMSLoaded:
00000BDA B80043                         mov     ax,4300h
00000BDD CD2F                           int     2Fh             ; - Multiplex - XMS - INSTALLATION CHECK
                                                                ; Return: AL = 80h XMS driver installed
                                                                ; AL <> 80h no driver
00000BDF 3C80                           cmp     al,80h          ; XMS installed?
00000BE1 C3                             retn

                                ; ---------------------------------------------------------------------------
                                ; procedure : FTryToMovDOSHi
                                ;
                                ;           Called from HMA suballoc calls
                                ;
                                ; ---------------------------------------------------------------------------

                                        ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                        ; (MSDOS 5.0 IO.SYS - SYSINIT:0A84h)

                                        ; 08/04/2024 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
                                        ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:0C9Fh)

                                        ; ((MSDOS 6.22 IO.SYS - SYSINIT:0B8Ch))

                                FTryToMovDOSHi:    ; proc  far

00000BE2 50                             push    ax
00000BE3 53                             push    bx
00000BE4 51                             push    cx
00000BE5 52                             push    dx
00000BE6 56                             push    si
00000BE7 57                             push    di
00000BE8 1E                             push    ds
00000BE9 06                             push    es

                                        ; 23/10/2022
                                        ; 27/03/2019 - Retro DOS v4.0
                                        ; 11/12/2022
00000BEA 0E                             push    cs
00000BEB 1F                             pop     ds

                                        ;cmp    byte [cs:runhigh],0FFh
                                        ; 11/12/2022
00000BEC 803E[6C02]FF                   cmp     byte [runhigh],0FFh
00000BF1 7503                           jne     short _ftymdh_1

                                        ; ds = cs
00000BF3 E8A2FE                         call    TryToMovDOSHi
                                _ftymdh_1:
00000BF6 07                             pop     es
00000BF7 1F                             pop     ds
00000BF8 5F                             pop     di
00000BF9 5E                             pop     si
00000BFA 5A                             pop     dx
```

```
00000BFB 59                             pop     cx
00000BFC 5B                             pop     bx
00000BFD 58                             pop     ax

00000BFE CB                             retf

                                ; ----------------------------------------------------------------------
                                ;
                                ; following piece of code will be moved into a para boundary. And the para
                                ; address posted in seg of int 19h vector. Offset of int 19h will point to
                                ; VDint19. This is to protect HMA from apps which use VDISK header method
                                ; to determine free extended memory.
                                ;
                                ; For more details read "power programming" column by Ray Duncan in the
                                ; May 30 1989 issue of PC Magazine (pp 377-388) [USING EXTENDED MEMORY,PART 1]
                                ;
                                ; ----------------------------------------------------------------------

                                    ; 30/12/2023 - Retro DOS 5.0
00000BFF 00                         db      0

                                ; 13/04/2024
                                ;align 2

                                    ; 30/12/2023
                                    ; PCDOS v7.1 IBMBIO.COM, SYSYINIT:0CBCh

                                StartVDHead:
                                ;-------------- what follows is a dummy device driver header (not used by DOS)

00000C00 00000000                   dd      0               ; link to next device driver
00000C04 0080                       dw      8000h           ; device attribute
00000C06 0000                       dw      0               ; strategy routine offset
00000C08 0000                       dw      0               ; interrupt routine offset
00000C0A 01                         db      1               ; number of units
                                    ;db     7 dup(0)
00000C0B 00<rep 7h>                 times   7 db 0          ; reserved area

                                VDiskSig1:
00000C12 564449534B                 db      'VDISK'

                                VLEN1       equ     ($-VDiskSig1)

00000C17 202056332E33               db      '  V3.3'        ; vdisk label
                                    ;db     15 dup (0)      ; pad
00000C1D 00<rep Fh>                 times   15 db 0
00000C2C 0000                       dw      0               ; bits 0-15 of free HMA
00000C2E 11                         db      11h             ; bits 16-23 of free HMA (1M + 64K)

                                VDInt19:
00000C2F EA                         db      0EAh            ; jmp to old vector

                                OldVDInt19:
00000C30 00000000                   dd      0               ; Saved int 19 vector

                                EndVDHead: ; label byte

                                VDiskHMAHead:
00000C34 000000                     db      0,0,0           ; non-bootable disk

                                VDiskSig2:
00000C37 564449534B                 db      'VDISK'

                                VLEN2       equ     ($-VDiskSig2)

00000C3C 332E33                     db      '3.3'           ; OEM - signature
00000C3F 8000                       dw      128             ; number of bytes/sector
00000C41 01                         db      1               ; sectors/cluster
00000C42 0100                       dw      1               ; reserved sectors
00000C44 01                         db      1               ; number of FAT copies
00000C45 4000                       dw      64              ; number of root dir entries
00000C47 0002                       dw      512             ; number of sectors
00000C49 FE                         db      0FEh            ; media descriptor
00000C4A 0600                       dw      6               ; number of sectors/FAT
00000C4C 0800                       dw      8               ; sectors per track
00000C4E 0100                       dw      1               ; number of heads
00000C50 0000                       dw      0               ; number of hidden sectors
00000C52 4004                       dw      440h            ; Start of free HMA in K (1M+64K)

                                EndVDiskHMAHead: ; label byte

                                ; ----------------------------------------------------------------------
                                ;
                                ; procedure : InstVDiskHeader
                                ;
                                ;             Installs the VDISK header to reserve the 64k of HMA
                                ;         It puts a 32 byte header at 10000:0 and
                                ;         another header at (seg of int19):0
                                ;
                                ; Inputs : None
                                ;
                                ; Outputs : None
                                ;
                                ; USES : DS,SI,AX,CX,DX
                                ;
                                ; ----------------------------------------------------------------------

                                    ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)

                                InstVDiskHeader:
00000C54 31C0                       xor     ax,ax
00000C56 8ED8                       mov     ds,ax               ; seg of int vect table

                                ;-------------- save old int 19 vector

                                    ; 23/10/2022
00000C58 A16400                     mov     ax,[19h*4]
                                    ;mov    [OldVDInt19],ax
00000C5B 2EA3[300C]                 mov     [cs:OldVDInt19],ax
00000C5F A16600                     mov     ax,[19h*4+2]
                                    ;mov    [OldVDInt19+2],ax
00000C62 2EA3[320C]                 mov     [cs:OldVDInt19+2],ax

                                ;-------------- calculate seg of new int 19 handler

00000C66 B448                       mov     ah,48h              ; allocate memory
                                    ;mov    bx,(EndVDHead-StartVDHead+15)>>4
                                    ; 23/10/2022
00000C68 BB0400                     mov     bx,4
00000C6B CD21                       int     21h

                                ;   if carry, fatal hanging error!!!!!

00000C6D 48                         dec     ax                  ; point to arena
00000C6E 8EC0                       mov     es,ax
                                    ;mov    word [es:ARENA.OWNER],8      ; owner = System
00000C70 26C70601000800             mov     word [es:1],8
                                    ;mov    word [es:ARENA.NAME],'SC' ; System Code
```

```
00000C77 26C70608005343             mov     word [es:8],'SC' ; 4353h
00000C7E 40                         inc     ax
00000C7F 8EC0                       mov     es,ax                   ; get back to allocated memory

                                ;-------------- install new int 19 vector

00000C81 FA                         cli                             ; no reboots at this time
                                    ;mov    word [19h*4],(VDInt19-StartVDHead)
00000C82 C70664002F00               mov     word [19h*4],47
00000C88 A36600                     mov     [19h*4+2],ax

                                ;-------------- move the code into proper place

                                    ;mov    cx,(EndVDHead-StartVDHead)
00000C8B B93400                     mov     cx,52
00000C8E BE[000C]                   mov     si,StartVDHead
00000C91 31FF                       xor     di,di
00000C93 0E                         push    cs
00000C94 1F                         pop     ds
00000C95 FC                         cld
00000C96 F3A4                       rep     movsb
00000C98 FB                         sti                             ; BUGBUG is sti OK now?

                                ;-------------- mov the HMA VDisk head into HMA

                                    ; 23/10/2022
00000C99 57                         push    di
00000C9A 06                         push    es

                                    ;mov    ax,0FFFFh
                                    ;mov    es,ax
                                    ; 03/09/2023
00000C9B 49                         dec     cx
                                    ; cx = 0FFFFh
00000C9C 8EC1                       mov     es,cx

00000C9E BF1000                     mov     di,10h
                                    ;mov    cx,(EndVDiskHMAHead-VDiskHMAHead)
00000CA1 B92000                     mov     cx,32
00000CA4 BE[340C]                   mov     si,VDiskHMAHead
00000CA7 F3A4                       rep     movsb                   ; ds already set to cs

00000CA9 5F                         pop     di
00000CAA 07                         pop     es

00000CAB C3                         retn

                                ; ---------------------------------------------------------------------------
                                ; procedure : ClrVDISKHeader
                                ;
                                ;           Clears the first 32 bytes at 1MB boundary
                                ;           So that DOS/HIMEM is not confused about the VDISK header
                                ;           left by previous DOS=HIGH session
                                ;
                                ; ---------------------------------------------------------------------------

                                struc desc
00000000 ????                    .seg_lim:  resw    1               ; seg limit 64K
00000002 ????                    .lo_word:  resw    1               ; 24 bit seg physical
00000004 ??                      .hi_byte:  resb    1               ; address
00000005 ??                      .acc_rights:       resb    1               ; access rights ( CPL0 - R/W )
00000006 ????                    .reserved: resw    1               ;
                                 .size:
                                endstruc

                                            ; 23/10/2022
                                bmove:              ;label byte

                                dummy:              ;times desc.size db 0 ; desc <>
00000CAC 00<rep 8h>                         times 8 db 0
                                gdt:        ;times desc.size db 0 ; desc <>
00000CB4 00<rep 8h>                         times 8 db 0
00000CBC FFFF                   src_desc:   dw      0FFFFh          ; desc <0ffffh,0,0,93h,0>
00000CBE 0000                               dw      0
00000CC0 00                                 db      0
00000CC1 93                                 db      93h
00000CC2 0000                               dw      0
00000CC4 FFFF                   tgt_desc:   dw      0FFFFh          ; desc <0ffffh,0,10h,93h,0>  ; 1MB
00000CC6 0000                               dw      0
00000CC8 10                                 db      10h
00000CC9 93                                 db      93h
00000CCA 0000                               dw      0

                                rombios_code:       ;times desc.size db 0 ; desc <>
00000CCC 00<rep 8h>                         times 8 db 0
                                temp_stack: ;times desc.size db 0 ; desc <>
00000CD4 00<rep 8h>                         times 8 db 0

00000CDC 00<rep 20h>            ClrdVDISKHead:      times 32 db 0           ; db 32 dup (0)


                                ; 25/03/2019 - Retro DOS v4.0 (MSDOS 6.21 IO.SYS, MSDOS 6.0 SYSINIT1.ASM)

                                ; 30/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:0CA6h)

                                ClrVDISKHeader:     ; proc  near

                                ;; 04/07/2023 - Retro DOS v4.2 IO.SYS (optimized)
                                ;;-------------------------------------------------------         ;IO70
                                ;; The following workaround get around a problem with the          ;IO70
                                ;; Tortugas and PS/2 30-286 BIOS when password server mode         ;IO70
                                ;; is set. On those machines the INT 15h block move code           ;IO70
                                ;; goes through the 8042 to twiddle A20 instead of port 92h.       ;IO70
                                ;; In password server mode the 8042 is disabled so the block       ;IO70
                                ;; move crashes the system. We can do this because these           ;IO70
                                ;; systems clear all of memory on a cold boot.                     ;IO70
                                ;                                                       ;IO70
                                ;               in      al,64h          ; Test for password servr mode ;IO70
                                ;               test    al,10h          ; Is keyboard inhibited?      ;IO70
                                ;               jnz     short ClrVDISKok ; No, go do block move.      ;IO70
                                ;                                       ; Check for Tortugas...       ;IO70
                                ;               cmp     word [cs:sys_model_byte],19F8h                ;IO70
                                ;               je      short ClrVDISKno                              ;IO70
                                ;                                       ; Check for mod 30-286        ;IO70
                                ;               cmp     word [cs:sys_model_byte],09FCh                ;IO70
                                ;               jne     short ClrVDISKok                ;IO70
                                ;ClrVDISKno:      retn                  ; Return w/o block move.            ;IO70
                                ;                                                       ;IO70
                                ;ClrVDISKok:                                                    ;IO70
                                ;-------------------------------------------------------         ;IO70

                                    ; 30/12/2023 - Retro DOS v5.0
                                    ; PCDOS 7.1 IBMBIO.COM - SYSINIT:0DBAh
```

```
                                ClrVDISKHeader:
00000CFC E464                       in      al,64h  ; 8042 keyboard controller status register
                                                    ; 7:  PERR    1=parity error in data received from keyboard
                                                    ;    +---------- AT Mode ----------+------------ PS/2 Mode ------------+
                                                    ; 6: |RxTO    receive (Rx) timeout  | TO     general timeout (Rx or Tx) |
                                                    ; 5: |TxTO    transmit (Tx) timeout | MOBF   mouse output buffer full   |
                                                    ;    +------------------------------+-----------------------------------+
                                                    ; 4:  INH     0=keyboard communications inhibited
                                                    ; 3:  A2      0=60h was the port last written to, 1=64h was last
                                                    ; 2:  SYS     distinguishes reset types: 0=cold reboot, 1=warm reboot
                                                    ; 1:  IBF     1=input buffer full (keyboard can't accept data)
                                                    ; 0:  OBF     1=output buffer full (data from keyboard is available)
00000CFE A810                       test    al,10h  ; test bit 4 - Is keyboard inhibited?
00000D00 7511                       jnz     short ClrVDISKok ; No, go do block move
                                    ; 30/12/2023
                                    ; ds = cs
00000D02 813E[BB02]F819             cmp     word [sys_model_byte],19F8h ; check for TORTUGA models
00000D08 7408                       jz      short ClrVDISKno ; do not use INT 15h block move code
                                                             ; (while 8042 is disabled)
00000D0A 813E[BB02]FC09             cmp     word [sys_model_byte],9FCh ; check for PS/2 30-286 model
00000D10 7501                       jnz     short ClrVDISKok
                                ClrVDISKno:
00000D12 C3                         retn
                                ; ---------------------------------------------------------------------------
                                    ; 30/12/2023
                                ClrVDISKok:
                                    ; 12/12/2022
                                    ; ds = cs

                                    ; 30/12/2022 - Retro DOS v4.2
                                    ; (MSDOS 6.21 IO.SYS SYSINIT:0CBFh)

00000D13 06                         push    es
00000D14 8CC8                       mov     ax,cs
00000D16 89C2                       mov     dx,ax
00000D18 B10C                       mov     cl,12
00000D1A D3EA                       shr     dx,cl
00000D1C B104                       mov     cl,4
00000D1E D3E0                       shl     ax,cl
00000D20 05[DC0C]                   add     ax,ClrdVDISKHead
00000D23 80D200                     adc     dl,0

                                    ;; 23/10/2022
                                    ;;mov   [cs:src_desc+desc.lo_word],ax
                                    ;mov    [cs:src_desc+2],ax
                                    ;;mov   [cs:src_desc+desc.hi_byte],dl
                                    ;mov    [cs:src_desc+4],dl
                                    ; 12/12/2022
                                    ;mov    [src_desc+desc.lo_word],ax
00000D26 A3[BE0C]                   mov     [src_desc+2],ax
                                    ;mov    [src_desc+desc.hi_byte],dl
00000D29 8816[C00C]                 mov     [src_desc+4],dl

00000D2D B91000                     mov     cx,16   ; 16 words
00000D30 0E                         push    cs
00000D31 07                         pop     es
00000D32 BE[AC0C]                   mov     si,bmove
00000D35 B487                       mov     ah,87h
00000D37 CD15                       int     15h     ; EXTENDED MEMORY - BLOCK MOVE (AT,XT286,PS)
                                                    ; CX = number of words to move
                                                    ; ES:SI -> global descriptor table
                                                    ; Return: CF set on error, AH = status
00000D39 07                         pop     es
00000D3A C3                         retn

                                ; ---------------------------------------------------------------------------
                                ;
                                ; procedure : SaveFreeHMAPtr
                                ;
                                ;           Save the Free HMA pointer in BIOS variable for later use.
                                ;           (INT 2f ax==4a01 call returns pointer to free HMA)
                                ;           Normalizes the pointer to ffff:xxxx format and stores only
                                ;           the offset.
                                ;
                                ; Inputs : ES:DI - pointer to free HMA
                                ; Output : FreeHMAPtr in BIOS data segment updated
                                ;
                                ; ---------------------------------------------------------------------------

                                SaveFreeHMAPtr:
                                    ; 03/09/2023
00000D3B 1E                         push    ds
00000D3C B87000                     mov     ax,DOSBIODATASEG ; 0070h
00000D3F 8ED8                       mov     ds,ax
                                    ;
00000D41 8CC3                       mov     bx,es
00000D43 B8FFFF                     mov     ax,0FFFFh         ; HMA segment
                                    ; 03/09/2023
00000D46 A2[0D00]                   mov     [inHMA],al ; 0FFh  ; (BIOSDATA:000Dh) ; 08/04/2024
                                    ;
00000D49 29D8                       sub     ax,bx
00000D4B 83C70F                     add     di,15             ; para round
00000D4E 83E7F0                     and     di,0FFF0h
00000D51 B104                       mov     cl,4
00000D53 D3E0                       shl     ax,cl
00000D55 29C7                       sub     di,ax
                                    ;
                                    ; 03/09/2023
                                    ;push   ds
                                    ;;mov   ax,Bios_Data ; 0070h
                                    ;mov    ax,KERNEL_SEGMENT ; 0070h
                                    ; 21/10/2022
                                    ; 03/09/2023
                                    ;mov    ax,DOSBIODATASEG ; 0070h
                                    ;mov    ds,ax
                                            ; (BIOSDATA:07D7h for PCDOS 7.1 IBMBIO.COM) ; 08/04/2024
00000D57 893E[D707]                 mov     [FreeHMAPtr],di           ; (ds:8F7h for MSDOS 6.21 IO.SYS)
                                    ;mov    byte [inHMA],0FFh  ; (ds:0Dh)
00000D5B 1F                         pop     ds
00000D5C C3                         retn

                                ; ---------------------------------------------------------------------------
                                ;
                                ; procedure : IsVDiskInstalled
                                ;
                                ;           Checks for the presence of VDISK header at 1MB boundary
                                ;           & INT 19 vector
                                ;
                                ; Inputs  : A20 flag should be ON
                                ; Outputs : Zero set if VDISK header found else Zero cleared
                                ;
                                ; ---------------------------------------------------------------------------

                                IsVDiskInstalled:
```

```
00000D5D 31C0                   xor     ax,ax
00000D5F 8ED8                   mov     ds,ax
00000D61 8E1E4E00               mov     ds,[19*4+2]
                                ;mov    si,VDiskSig1-StartVDHead ; 12h
                                ; 23/10/2022
00000D65 BE1200                 mov     si,12h ; 18
                                ;mov    cx,VLEN1 ; 5
00000D68 B90500                 mov     cx,5
00000D6B 0E                     push    cs
00000D6C 07                     pop     es
00000D6D BF[120C]               mov     di,VDiskSig1
00000D70 F3A6                   rep     cmpsb
00000D72 740F                   je      short ivdins_retn
00000D74 B8FFFF                 mov     ax,0FFFFh
00000D77 8ED8                   mov     ds,ax
                                ;mov    si,10h+(VDiskSig2-VDiskHMAHead) ; 13h
00000D79 BE1300                 mov     si,13h
00000D7C BF[370C]               mov     di,VDiskSig2
                                ;;mov   cx,VLEN2  ; 5
                                ;mov    cx,5
                                ; 03/09/2023
00000D7F B105                   mov     cl,5
00000D81 F3A6                   rep     cmpsb
                        ivdins_retn:
00000D83 C3                     retn                    ; returns the Zero flag

                        ; ---------------------------------------------------------------------------
                        ;
                        ; procedure : CPMHack
                        ;
                        ;           Copies the code from 0:c0 into ffff:0d0h
                        ;           for CPM compatibility
                        ;
                        ; ---------------------------------------------------------------------------

                                ; 11/12/2022
                        CPMHack:
00000D84 1E                     push    ds
00000D85 B9FFFF                 mov     cx,0FFFFh
00000D88 8EC1                   mov     es,cx           ; ES = FFFF
                                ;xor    cx,cx
                                ; 11/12/2022
00000D8A 41                     inc     cx  ; cx = 0
00000D8B 8ED9                   mov     ds,cx           ; DS = 0
00000D8D BEC000                 mov     si,0C0h
00000D90 BFD000                 mov     di,0D0h
                                ;mov    cx,5
00000D93 B105                   mov     cl,5
00000D95 FC                     cld
00000D96 F3A4                   rep     movsb           ; move 5 bytes from 0:C0 to FFFF:D0
00000D98 1F                     pop     ds
00000D99 C3                     retn

                        ; ---------------------------------------------------------------------------
                        ;
                        ; procedure : off_to_para
                        ;
                        ; ---------------------------------------------------------------------------
                        off_to_para:
00000D9A D1E8                   shr     ax,1
00000D9C D1E8                   shr     ax,1
00000D9E D1E8                   shr     ax,1
00000DA0 D1E8                   shr     ax,1
00000DA2 C3                     retn

                        ; ---------------------------------------------------------------------------
                        ;** TempCDS - Create (Temporary?) CDS
                        ;
                        ;   ENTRY   ?? BUGBUG
                        ;           (DS) = SysInitSeg
                        ;   EXIT    ?? BUGBUG
                        ;   USES    ?? BUGBUG
                        ; ---------------------------------------------------------------------------

                                ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ; 30/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; 30/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
                        TempCDS:
00000DA3 C43E[6D02]             les     di,[DOSINFO]
00000DA7 268A4D20               mov     cl,[es:di+SYSI_NUMIO]

                                ;mov    cl,[es:di+20h]
00000DAB 30ED                   xor     ch,ch                   ; (cx) = # of block devices

00000DAD 26884D21               mov     [es:di+SYSI_NCDS],cl  ; one CDS per device
                                ;mov    [es:di+21h],cl

                                ;mov    al,cl
                                ;mov    ah,curdirlen ; curdir_list.size ; 88
                                ;;mov   ah,88
                                ;mul    ah                      ; (ax) = byte size for those CDSs
                                ; 30/12/2023
00000DB1 B058                   mov     al,curdirlen ; curdir_list.size ; 88
                                ;mov    al,88
00000DB3 F6E1                   mul     cl                      ; (ax) = byte size for those CDSs

00000DB5 E85D05                 call    ParaRound               ; (ax) = paragraph size for CDSs
00000DB8 8B36[A702]             mov     si,[top_of_cdss] ; 31/12/2022

                        ;   BUGBUG - we don't update confbot - won't someone else use it?
                        ;   chuckst -- answer: no. Confbot is used to access the CDSs,
                        ;   25 jul 92  which are stored BELOW it. Alloclim is the
                        ;              variable which has the top of free memory for
                        ;              device driver loads, etc.

00000DBC 29C6                   sub     si,ax

                        ;   chuckst, 25 Jul 92 -- note: I'm removing the code here
                        ;          that automatically updates alloclim every time we
                        ;          set up some new CDSs. Instead, I've added code
                        ;          which pre-allocates space for 26 CDSs. This
                        ;           way we've got room for worst case CDSs before
                        ;          we place MagicDrv.sys
                        ;
                        ;   mov     [ALLOCLIM],si         ; can't alloc past here!

                                ; 30/12/2022
                                ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ; (SYSINIT:0C52h)
                                ;mov    [ALLOCLIM],si ; (MSDOS 5.0 SYSINIT)

00000DBE 26897518               mov     [es:di+SYSI_CDS+2],si
                                ;mov    [es:di+18h],si
00000DC2 89F0                   mov     ax,si
```

```
00000DC4 26C745160000               mov     word [es:di+SYSI_CDS],0      ; set address of CDS list
                                    ;mov    [word es:di+16h],0
                                    ;lds    si,[es:di+SYSI_DPB]   ; (ds:si) = address of first DPB
00000DCA 26C535                     lds     si,[es:di]
00000DCD 8EC0                       mov     es,ax
00000DCF 31FF                       xor     di,di                 ; (es:di) = address of 1st CDS

                                ;*  Initialize our temporary CDSs. We'll init each CDS with the
                                ;   info from the corresponding DPB.
                                ;
                                ;   (cx) = count of CDSs left to process
                                ;   (es:di) = address of next CDS

                                fooset:
                                    ; 23/10/2022
00000DD1 2EA1[A902]                 mov     ax,[cs:DirStrng] ; "A:"
00000DD5 AB                         stosw                         ; setup the root as the curdir

                                    ; 23/10/2022 (MSDOS 5.0 SYSINIT)
                                    ;call   get_dpb_for_drive_al  ; get dpb for drive in dpb

                                    ; 30/12/2022
                                    ; (MSDOS 6.21 SYSINIT:0D8Bh)
00000DD6 E85200                     call    get_dpb_for_drive_al  ; get dpb for drive in dpb

                                ;   (ds:si) = address of DPB
                                ;            (si) = -1 if no drive

00000DD9 2EA1[AB02]                 mov     ax,[cs:DirStrng+2] ; "\",0
00000DDD AB                         stosw
00000DDE 2EFE06[A902]               inc     byte [cs:DirStrng]
00000DE3 31C0                       xor     ax,ax ; 0
00000DE5 51                         push    cx
                                    ;mov    cx,curdir_list.cdir_flags - 4 ; 63
00000DE6 B93F00                     mov     cx,63   ; 23/10/2022
00000DE9 F3AA                       rep     stosb                 ; zero out rest of CURDIR_TEXTs

                                ;   should handle the system that does not have any floppies.
                                ;   in this case,we are going to pretended there are two dummy floppies
                                ;   in the system. still they have dpb and cds,but we are going to
                                ;   0 out curdir_flags,curdir_devptr of cds so ibmdos can issue
                                ;   "invalid drive specification" message when the user try to
                                ;   access them.
                                ;
                                ;   (ax) = 0
                                ;   (es:di) = CURDIR_FLAGS in the CDS records
                                ;   (ds:si) = Next DPB (-1 if none)

00000DEB 83FEFF                     cmp     si,-1   ; cmp si,0FFFFh
00000DEE 740C                       je      short fooset_zero     ; don't have any physical drive.

                                ;   check to see if we are faking floppy drives. if not go to normcds.
                                ;   if we are faking floppy drives then see if this cds being initialised
                                ;   is for drive a: or b: by checking the appropriate field in the dpb
                                ;   pointed to by ds:si. if not for a: or b: then go to normcds. if
                                ;   for a: or b: then execute the code given below starting at fooset_zero.
                                ;   for dpb offsets look at inc\dpb.inc.

                                    ; 03/09/2023
00000DF0 41                         inc     cx  ; cx = 1

00000DF1 2E380E[8B02]               cmp     [cs:fake_floppy_drv],cl ; 1 ; 03/09/2023
                                    ;cmp    byte [cs:fake_floppy_drv],1
00000DF6 750A                       jne     short normcds         ; machine has floppy drives
                                    ;cmp    byte [si+DPB.drive],1 ; if dpb_drive = 0 (a) or 1 (b).
                                    ;cmp    byte [si],1
00000DF8 380C                       cmp     [si],cl ; 1 ; 03/09/2023
00000DFA 7706                       ja      short normcds

                                    ; 30/12/2023
                                    ; ax = 0
                                fooset_zero:
00000DFC B103                       mov     cl,3                  ; the next dbp pointer
                                                                  ; AX should be zero here
00000DFE F3AB                       rep     stosw
                                    ; 30/12/2023
                                    ;pop    cx
00000E00 EB0F                       jmp     short get_next_dpb ; findcds

                                ;   (ax) = 0

                                    ; 30/12/2023
                                ;fooset_zero:
                                    ;mov    cl,3
                                    ;rep    stosw
                                    ;pop    cx
                                    ;jmp    short fincds

                                ;*  We have a "normal" DPB and thus a normal CDS.
                                ;
                                ;   (ax) = 0
                                ;   (es:di) = CURDIR_FLAGS in the CDS records
                                ;   (ds:si) = Next DPB (-1 if none)

                                normcds:
                                    ; 30/12/2023
                                    ;pop    cx

                                ;   if a non-fat based media is detected (by dpb.numberoffat == 0), then
                                ;   set curdir_flags to 0. this is for signaling ibmdos and ifsfunc that
                                ;   this media is a non-fat based one.

                                    ;cmp    byte [si+DPB.FAT_COUNT],0 ; non fat system?
                                    ; 23/10/2022
                                    ;cmp    byte [si+8],0
                                    ; 03/09/2023 (ax=0)
00000E02 384408                     cmp     [si+8],al ; 0
00000E05 7403                       je      short setnormcds      ; yes. set curdir_flags to 0. ax = 0 now.
00000E07 B80040                     mov     ax,curdir_inuse ; 4000h       ; else,fat system. set the flag to curdir_inuse.
                                    ;mov    ax,4000h
                                setnormcds:
00000E0A AB                         stosw                         ; curdir_flags
00000E0B 89F0                       mov     ax,si
00000E0D AB                         stosw                         ; curdir_devptr
00000E0E 8CD8                       mov     ax,ds
00000E10 AB                         stosw

                                get_next_dpb:                             ; entry point for fake_fooset_zero
                                    ; 30/12/2022
                                    ; (MSDOS 6.21 SYSINIT:0DD1h)
                                    ; 23/10/2022
                                    ;lds    si,[si+19h] ; (MSDOS 5.0 SYSINIT)
                                    ;;lds   si,[si+DPB.NEXT_DPB] ; [si+19h]
                                fincds:     ; get_next_dpb
```

```
                                    ; 30/12/2023
00000E11 59                         pop     cx
                                    ; 30/12/2022
                                    ; (MSDOS 6.21 SYSINIT:0DD1h)
00000E12 B8FFFF                     mov     ax,-1   ; mov ax,0FFFFh
00000E15 AB                         stosw                           ; curdir_id
00000E16 AB                         stosw                           ; curdir_id
00000E17 AB                         stosw                           ; curdir_user_word
00000E18 B80200                     mov     ax,2
00000E1B AB                         stosw                           ; curdir_end
00000E1C B000                       mov     al,0                    ; clear out 7 bytes (curdir_type,
00000E1E AA                         stosb
00000E1F AB                         stosw                           ;  curdir_ifs_hdr,curdir_fsda)
00000E20 AB                         stosw
00000E21 AB                         stosw

00000E22 E2AD                       loop    fooset

00000E24 2EC606[A902]41             mov     byte [cs:DirStrng],"A"; "A:\"

00000E2A C3                         retn

                                ; ---------------------------------------------------------------------------
                                ;***get_dpb_for_drive_al -- lookup the DPB for drive in al
                                ;
                                ;   entry:
                                ;      al == ASCII CAPS drive letter
                                ;
                                ;   exit:
                                ;      ds:si -> DPB, or si = -1 if not found
                                ; ---------------------------------------------------------------------------

                                ; 30/12/2023
                                ; PCDOS 7.1 IBMBIO.COM - SYSINIT:0EFEh

                                    ; 30/12/2022
                                    ; (MSDOS 6.21 SYSINIT:0DEAh)
                                    ; 23/10/2022
                                get_dpb_for_drive_al:
00000E2B 2EC536[6D02]               lds     si,[cs:DOSINFO]             ; point to first DPB
                                    ;lds    si,[si+SYSI_DPB]        ; (ds:si) = address of first DPB
00000E30 C534                       lds     si,[si]
00000E32 2C41                       sub     al,'A'

                                get_dpb_for_drive_1:
                                    ;cmp    al,[si+DPB.DRIVE]       ; match?
00000E34 3A04                       cmp     al,[si]
00000E36 7408                       je      short got_dpb_for_drive     ;  done if so

00000E38 C57419                     lds     si,[si+DPB.NEXT_DPB] ; [si+19h]
00000E3B 83FEFF                     cmp     si,-1
00000E3E 75F4                       jne     short get_dpb_for_drive_1 ; loop until hit end of DPBs

                                got_dpb_for_drive:
00000E40 C3                         retn

                                ;============================================================================

                                ;** EndFile - Build DOS structures
                                ;
                                ; This procedure is called after the config.sys has been processed and
                                ; installable device drivers have been loaded (but before "install="
                                ; programs are loaded) to create the dos structures such as SFTs, buffers,
                                ; FCBs, CDSs, etc. It also loads the sysinit_base module in low memory
                                ; to allow for the safe EXECing of "install=" programs. All memory
                                ; above these structures is deallocated back to DOS.
                                ;
                                ;   ENTRY   ?? BUGBUG
                                ;   EXIT    ?? BUGBUG
                                ;   USES    ?? BUGBUG

                                ;============================================================================
                                ; allocate files
                                ; ---------------------------------------------------------------------------

                                    ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:0CCDh)

                                    ; 30/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:0E00h)

                                    ; 09/04/2024 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:0F14h)

                                    ; ((MSDOS 6.22 IO.SYS - SYSINIT:0E00h))

                                endfile:
                                ; we are now setting up final cdss,buffers,files,fcss strings etc. we no
                                ; longer need the space taken by the temp stuff below confbot,so set alloclim
                                ; to confbot.

                                ;   if this procedure has been called to take care of install= command,
                                ;      then we have to save es,si registers.

                                    ; 11/12/2022
                                    ; ds = cs

                                    ; 23/10/2022
                                    ; 31/03/2019
00000E41 1E                         push    ds

                                    ;;mov   ax,Bios_Data ; 0070h
                                    ;mov    ax,KERNEL_SEGMENT ; 0070h
                                    ; 21/10/2022
00000E42 B87000                     mov     ax,DOSBIODATASEG ; 0070h
00000E45 8ED8                       mov     ds,ax

                                    ;cmp    word [052Fh],0
00000E47 833E[A004]00               cmp     word [multrk_flag],multrk_off1 ;=0,multrack= command entered?
00000E4C 7505                       jne     short multrk_flag_done
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;or     word [multrk_flag],multrk_on ; 80h  ; default will be on.
                                    ; 12/12/2022
00000E4E 800E[A004]80               or      byte [multrk_flag],multrk_on ; 80h
                                multrk_flag_done:
                                    ; 23/10/2022
                                    ; 31/03/2019
00000E53 1F                         pop     ds

                                    ; 11/12/2022
                                    ; ds = cs
                                    ;mov    ax,[top_of_cdss] ; mov ax,[CONFBOT]
                                    ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:0E14h)
```

```
00000E54 A1[A302]                   mov     ax,[CONFBOT]
00000E57 A3[A502]                   mov     [ALLOCLIM],ax
                                    ; 23/10/2022
                                    ;mov    ax, [cs:top_of_cdss]
                                    ;mov    [cs:ALLOCLIM], ax

                                    ; 11/12/2022
                                    ; ds = cs
                                    ;push   cs
                                    ;pop    ds

                                    ;mov    ax,[CONFBOT]
                                    ;mov    [ALLOCLIM],ax

                                    ; 09/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                    ;;;
                                    ;;mov   ax,[cs:ALLOCLIM]
                                    ;mov    ax,[ALLOCLIM]
                                    ;mov    [cs:prev_alloclim],ax
00000E5A A3[6C03]                   mov     [prev_alloclim],ax
                                    ;mov    ax,[cs:memhi]
00000E5D A1[6403]                   mov     ax,[memhi]
                                    ;mov    [cs:prev_memhi],ax
00000E60 A3[6A03]                   mov     [prev_memhi],ax
                                dosfts:
                                    ;;;

00000E63 E88C39                     call    round

                                    ; 11/12/2022
                                    ; ds = cs
00000E66 A0[9F02]                   mov     al,[FILES]
                                    ; 23/10/2022
                                    ;mov    al,[cs:FILES]
00000E69 2C05                       sub     al,5
00000E6B 764B                       jbe     short dofcbs

00000E6D 50                         push    ax
                                    ;mov    al,devmark_files ; 'F'
00000E6E B046                       mov     al,'F'
00000E70 E81808                     call    setdevmark              ; set devmark for sfts (files)
00000E73 58                         pop     ax
00000E74 30E4                       xor     ah,ah                   ; do not use cbw instruction!!!!!
                                                                    ;  it does sign extend.
                                    ; 11/12/2022
                                    ; ds = cs
00000E76 8B1E[6203]                 mov     bx,[memlo]
00000E7A 8B16[6403]                 mov     dx,[memhi]
00000E7E C53E[6D02]                 lds     di,[DOSINFO]            ;get pointer to dos data
                                    ; 23/10/2022
                                    ;mov    bx,[cs:memlo]
                                    ;mov    dx,[cs:memhi]
                                    ;lds    di,[cs:DOSINFO]

                                    ;lds    di,[di+SYSI_SFT]        ;ds:bp points to sft
00000E82 C57D04                     lds     di,[di+4]

                                    ;mov    [di+SF.SFLink],bx
00000E85 891D                       mov     [di],bx
00000E87 895502                     mov     [di+SF.SFLink+2],dx     ;set pointer to new sft

00000E8A 0E                         push    cs
00000E8B 1F                         pop     ds

                                    ; 11/12/2022
                                    ; ds = cs
00000E8C C43E[6203]                 les     di,[memlo]              ;point to new sft
                                    ; 23/10/2022
                                    ;les    di,[cs:memlo]

                                    ;mov    word [es:di+SF.SFLink],-1
00000E90 26C705FFFF                 mov     word [es:di],-1                 ; 0FFFFh
                                    ;mov    [es:di+SF.SFCount],ax
00000E95 26894504                   mov     [es:di+4],ax
                                    ; 09/04/2024
00000E99 B33B                       mov     bl,SF_ENTRY.size ; 59
                                    ;mov    bl,59
00000E9B F6E3                       mul     bl                      ;ax = number of bytes to clear
00000E9D 89C1                       mov     cx,ax
                                    ; 11/12/2022
                                    ; ds = cs
00000E9F 0106[6203]                 add     [memlo],ax              ;allocate memory
                                    ; 23/10/2022
                                    ;add    [cs:memlo],ax
00000EA3 B80600                     mov     ax,6
                                    ; 11/12/2022
00000EA6 0106[6203]                 add     [memlo],ax              ;remember the header too
                                    ;add    [cs:memlo],ax
                                    ; 11/12/2022
00000EAA 800E[6919]02               or      byte [setdevmarkflag],for_devmark ; 2
                                    ; 23/10/2022
                                    ;or     byte [cs:setdevmarkflag],2
00000EAF E84039                     call    round                   ; check for mem error before the stosb
00000EB2 01C7                       add     di,ax
00000EB4 31C0                       xor     ax,ax
00000EB6 F3AA                       rep     stosb                   ;clean out the stuff

                                ; allocate fcbs
                                ; ---------------------------------------------------------------------

                                    ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:0D48h)
                                dofcbs:
                                    ; 11/12/2022
                                    ; ds = cs
                                    ;push   cs
                                    ;pop    ds
00000EB8 E83739                     call    round
                                    ;mov    al,devmark_fcbs         ; 'X'   ;='x'
00000EBB B058                       mov     al,'X'
00000EBD E8CB07                     call    setdevmark
                                    ; 11/12/2022
                                    ; ds = cs
00000EC0 A0[A002]                   mov     al,[FCBS]
                                    ;mov    al,[cs:FCBS]
00000EC3 30E4                       xor     ah,ah                   ; do not use cbw instruction!!!!!
                                                                    ;  it does sign extend.
                                    ; 11/12/2022
00000EC5 8B1E[6203]                 mov     bx,[memlo]
00000EC9 8B16[6403]                 mov     dx,[memhi]
00000ECD C53E[6D02]                 lds     di,[DOSINFO]            ;get pointer to dos data
                                    ; 23/10/2022
                                    ;mov    bx,[cs:memlo]
                                    ;mov    dx,[cs:memhi]
```

```
                                    ;lds    di,[cs:DOSINFO]

                                    ;mov    [di+SYSI_FCB],bx
                                    ;mov    [di+SYSI_FCB+2],dx ;set pointer to new table
                                    ; 23/10/2022
00000ED1 895D1A                     mov     [di+1Ah],bx             ; [di+SYSI_FCB]
00000ED4 89551C                     mov     [di+1Ch],dx             ; [di+SYSI_FCB+2]

00000ED7 2E8A1E[A102]               mov     bl,[cs:KEEP]
00000EDC 30FF                       xor     bh,bh
                                    ;mov    [di+SYSI_KEEP],bx
00000EDE 895D1E                     mov     [di+1Eh],bx             ; [di+SYSI_KEEP]

00000EE1 0E                         push    cs
00000EE2 1F                         pop     ds

00000EE3 C43E[6203]                 les     di,[memlo]              ;point to new table
                                    ;mov    word [es:di+SF.SFLink],-1
00000EE7 26C705FFFF                 mov     word [es:di],-1
                                    ;mov    [es:di+SF.SFCount],ax
                                    ; 02/11/2022
00000EEC 26894504                   mov     [es:di+4],ax
00000EF0 B33B                       mov     bl,SF_ENTRY.size ; 59
00000EF2 89C1                       mov     cx,ax
00000EF4 F6E3                       mul     bl                      ;ax = number of bytes to clear
00000EF6 0106[6203]                 add     [memlo],ax              ;allocate memory
                                    ;mov    ax,6
00000EFA B80600                     mov     ax,SF.size-2 ; 6
00000EFD 0106[6203]                 add     [memlo],ax              ;remember the header too
                                    ;or     byte [setdevmarkflag],for_devmark ; 2
00000F01 800E[6919]02               or      byte [setdevmarkflag],2
00000F06 E8E938                     call    round                   ; check for mem error before the stosb
00000F09 01C7                       add     di,ax                   ;skip over header
00000F0B B041                       mov     al,'A'
                                fillloop:
00000F0D 51                         push    cx                      ; save count
00000F0E B93B00                     mov     cx,SF_ENTRY.size ; 59 ; number of bytes to fill
00000F11 FC                         cld
00000F12 F3AA                       rep     stosb                   ; filled

                                    ;mov    word [es:di-(SF_ENTRY.size)+SF_ENTRY.sf_ref_count],0  ; [es:di-59]
                                    ;mov    word [es:di-(SF_ENTRY.size)+SF_ENTRY.sf_position],0   ; [es:di-38]
                                    ;mov    word [es:di-(SF_ENTRY.size)+SF_ENTRY.sf_position+2],0 ; [es:di-36]

                                    ; 18/12/2022
                                    ;cx = 0
00000F14 26894DC5                   mov     [es:di-(SF_ENTRY.size)+SF_ENTRY.sf_ref_count],cx ;0  ; [es:di-59]
00000F18 26894DDA                   mov     [es:di-(SF_ENTRY.size)+SF_ENTRY.sf_position],cx ;0   ; [es:di-38]
00000F1C 26894DDC                   mov     [es:di-(SF_ENTRY.size)+SF_ENTRY.sf_position+2],cx ;0 ; [es:di-36]

                                    ; 23/10/2022
                                    ;mov     word [es:di-3Bh],0
                                    ;mov     word [es:di-26h],0
                                    ;mov     word [es:di-24h],0

00000F20 59                         pop     cx
00000F21 E2EA                       loop    fillloop

                                ; allocate buffers
                                ; ---------------------------------------------------------------------------

                                ; search through the list of media supported and allocate 3 buffers if the
                                ; capacity of the drive is > 360kb

                                    ; 18/12/2022
                                    ; cx = 0
00000F23 833E[9902]FF               cmp     word [buffers],-1       ; has buffers been already set?
00000F28 7403                       je      short dodefaultbuff
00000F2A E98000                     jmp     dobuff                  ; the user entered the buffers=.

                                dodefaultbuff:
                                    ; 18/12/2022
00000F2D 890E[9B02]                 mov     [h_buffers],cx ; 0
                                    ;inc    cx
                                    ;inc    cx
                                    ;mov    [buffers],cx ; 2
                                    ; 10/04/2024
00000F31 C706[9902]0200             mov     word [buffers],2

                                    ;mov    word [h_buffers],0      ; default is no heuristic buffers.
                                    ;mov    word [buffers],2        ; default to 2 buffers

                                    ; 23/10/2022
                                    ; 04/09/2023
                                    ;push   ax
                                    ;push   ds ; 26/03/2019

                                    ; 04/09/2023
                                    ; ds = cs
00000F37 C42E[6D02]                 les     bp,[DOSINFO]            ; search through the dpb's
                                    ;les    bp,[cs:DOSINFO]
                                    ;les    bp,[es:bp+SYSI_DPB]     ; get first dpb
                                    ; 11/12/2022
00000F3B 26C46E00                   les     bp,[es:bp]
                                    ; 23/10/2022
                                    ;les    bp,[es:bp+0]    ; ! (MSDOS 5.0 IO.SYS address compability) !

                                    ; 04/09/2023
                                    ; ds = cs
                                    ;push   cs
                                    ;pop    ds
                                ;SYSINIT:0DE2h:
                                nextdpb:                            ; test if the drive supports removeable media
                                    ;mov    bl,[es:bp+DPB.drive]
                                    ; 11/12/2022
00000F3F 268A5E00                   mov     bl,[es:bp]
                                    ; 23/10/2022
                                    ;mov    bl,[es:bp+0]    ; ! (MSDOS 5.0 IO.SYS address compability) !

                                    ;inc    bl
                                    ; 18/12/2022
00000F43 43                         inc     bx

                                    ;mov    ax,(IOCTL<<8)|8
00000F44 B80844                     mov     ax,4408h
00000F47 CD21                       int     21h             ; DOS - 2+ - IOCTL -

                                ; ignore fixed disks

00000F49 09C0                       or      ax,ax                   ; ax is nonzero if disk is nonremoveable
00000F4B 7534                       jnz     short nosetbuf

                                ; get parameters of drive
```

```
00000F4D 31DB                           xor     bx,bx
                                        ;;mov   bl,[es:bp+DPB.drive]
                                        ; 11/12/2022
00000F4F 268A5E00                       mov     bl,[es:bp]
                                        ; 23/10/2022
                                        ;mov    bl,[es:bp+0]   ; ! (MSDOS 5.0 IO.SYS address compability) !

                                        ;inc    bl
                                        ; 18/12/2022
00000F53 43                             inc     bx

00000F54 BA[BE4D]                       mov     dx,deviceparameters
                                        ;mov    ax,(IOCTL<<8)|GENERIC_IOCTL
00000F57 B80D44                         mov     ax,440Dh
                                        ;mov    cx,(RAWIO<<8)|GET_DEVICE_PARAMETERS
00000F5A B96008                         mov     cx,860h
00000F5D CD21                           int     21h             ; DOS - 2+ - IOCTL -
00000F5F 7220                           jc      short nosetbuf          ; get next dpb if driver doesn't support
                                                                        ; generic ioctl
                                    ; determine capacity of drive
                                    ; media capacity = #sectors * bytes/sector

                                        ;mov    bx,[deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.TOTALSECTORS]
                                        ; 23/10/2022
00000F61 8B1E[CD4D]                     mov     bx,[deviceparameters+15] ; total sectors (16 bit)

                                    ; to keep the magnitude of the media capacity within a word,
                                    ; scale the sector size
                                    ; (ie. 1 -> 512 bytes,2 -> 1024 bytes,...)

                                        ;mov    ax,[deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.BYTESPERSECTOR]
                                        ; 23/10/2022
00000F65 A1[C54D]                       mov     ax,[deviceparameters+7] ; bytes per sector
00000F68 31D2                           xor     dx,dx
00000F6A B90002                         mov     cx,512
00000F6D F7F1                           div     cx                      ; scale sector size in factor of
                                                                        ; 512 bytes
00000F6F F7E3                           mul     bx                      ; ax = #sectors * size factor
00000F71 09D2                           or      dx,dx                   ; just in case of large floppies
00000F73 7505                           jnz     short setbuf
00000F75 3DD002                         cmp     ax,720                  ; 720 sectors * size factor of 1
00000F78 7607                           jbe     short nosetbuf
                                    setbuf:
                                        ; 18/12/2022
                                        ; word [buffers] = 2
00000F7A C606[9902]03                   mov     byte [buffers],3
                                        ;mov    word [buffers],3
00000F7F EB0D                           jmp     short chk_memsize_for_buffers ; now check the memory size
                                                                        ; for default buffer count
                                    nosetbuf:
                                        ; 23/10/2022
                                        ;cmp    word [es:bp+DPB.NEXT_DPB],-1
00000F81 26837E19FF                     cmp     word [es:bp+19h], -1 ; 0FFFFh
00000F86 7406                           je      short chk_memsize_for_buffers
                                        ;les    bp,[es:bp+DPB.NEXT_DPB] ; [es:bp+19h]
00000F88 26C46E19                       les     bp,[es:bp+19h]
00000F8C EBB1                           jmp     short nextdpb

                                    ;from dos 3.3,the default number of buffers will be changed according to the
                                    ;memory size too.
                                    ; default buffers = 2
                                    ; if diskette media > 360 kb,then default buffers = 3
                                    ; if memory size > 128 kb (2000h para),then default buffers = 5
                                    ; if memory size > 256 kb (4000h para),then default buffers = 10
                                    ; if memory size > 512 kb (8000h para),then default buffers = 15.

                                    chk_memsize_for_buffers:
                                        ; 18/12/2022
                                        ;cmp    word [MEMORY_SIZE],2000h
                                        ;jbe    short bufset
                                        ;mov    word [buffers],5
                                        ;cmp    word [MEMORY_SIZE],4000h
                                        ;jbe    short bufset
                                        ;mov    word [buffers],10
                                        ;cmp    word [MEMORY_SIZE],8000h
                                        ;jbe    short bufset
                                        ;mov    word [buffers],15

                                        ; 18/12/2022
                                        ; word [buffers] = 3 or 2
00000F8E BB[9902]                       mov     bx,buffers
00000F91 A1[9402]                       mov     ax,[MEMORY_SIZE]
00000F94 48                             dec     ax      ; [MEMORY_SIZE] - 1

00000F95 80FC20                         cmp     ah,20h  ; ax >= 2000h ([MEMORY_SIZE] > 2000h) ; *
00000F98 7213                           jb      short bufset
00000F9A C6070F                         mov     byte [bx],15 ; [buffers] = 15 ; ***
00000F9D 80FC80                         cmp     ah,80h  ; ax >= 8000h ([MEMORY_SIZE] > 8000h) ; ***
00000FA0 730B                           jnb     short bufset
00000FA2 C6070A                         mov     byte [bx],10 ; [buffers] = 10 ; **
00000FA5 80FC40                         cmp     ah,40h  ; ax >= 4000h ([MEMORY_SIZE] > 4000h) ; **
00000FA8 7303                           jnb     short bufset
00000FAA C60705                         mov     byte [bx],5  ; [buffers] = 5 ; *
                                    bufset:
                                        ; 23/10/2022
                                        ; 26/03/2019
                                        ; 04/09/2023
                                        ;pop    ds
                                        ;pop    ax

                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;j.k. here we should put extended stuff and new allocation scheme!!!
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                    ; 26/03/2019

                                    ;********************************************************************************
                                    ;                                                                               *
                                    ; function: actually allocate buffers in the memory and initialize it.                *
                                    ; input :                                                                        *
                                    ;    memhi:memlo - start of the next available memory                           *
                                    ;    buffers = number of buffers                                                *
                                    ;    h_buffers = number of secondary buffers                                    *
                                    ;                                                                               *
                                    ; output:                                                                       *
                                    ;   buffinfo.cache_count - # of caches to be installed.                        *
                                    ;   buffinfo set.                                                              *
                                    ;   bufferqueue set.                                                           *
                                    ;                                                                               *
                                    ; subroutines to be called:                                                     *
                                    ;                                                                               *
                                    ;********************************************************************************

                                        ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
```

```
                                    ; (SYSINIT:0E60h)
                                dobuff:
                                    ; ds = cs ; 31/03/2019
                                    ; 23/10/2022
                                    ;lds    bx,[cs:DOSINFO]       ; ds:bx -> sysinitvar
                                    ; 04/09/2023
00000FAD A1[9902]                   mov     ax,[buffers] ; 31/03/2019
00000FB0 8B0E[9B02]                 mov     cx,[h_buffers] ; *
00000FB4 C51E[6D02]                 lds     bx,[DOSINFO]
                                    ;mov    ax,[cs:buffers]       ; set sysi_buffers
                                    ;mov    [bx+SYSI_BUFFERS],ax ; [bx+3Fh]
00000FB8 89473F                     mov     [bx+3Fh],ax
                                    ; 04/09/2023
                                    ;mov    ax,[cs:h_buffers]
                                    ;;mov   [bx+SYSI_BUFFERS+2],ax ; [bx+41h]
                                    ;mov    [bx+41h],ax
                                    ; 04/09/2023
00000FBB 894F41                     mov     [bx+41h],cx ; *
00000FBE C55F12                     lds     bx,[bx+12h]
                                    ;lds    bx,[bx+SYSI_BUF] ; now,ds:bx -> buffinfo
00000FC1 E82E38                     call    round           ; get [memhi]:[memlo]
                                    ;mov    al,devmark_buf ; ='B'
00000FC4 B042                       mov     al,'B'
00000FC6 E8C206                     call    setdevmark

                                ;allocate buffers

00000FC9 1E                         push    ds                    ; save buffer info. ptr.
00000FCA 53                         push    bx

00000FCB E8D403                     call    set_buffer

00000FCE 5B                         pop     bx
00000FCF 1F                         pop     ds

                                ;now set the secondary buffer if specified.

00000FD0 2E833E[9B02]00             cmp     word [cs:h_buffers],0
00000FD6 742D                       je      short xif16
00000FD8 E81738                     call    round
                                    ; 23/10/2022
00000FDB 2E8B0E[6203]               mov     cx,[cs:memlo]
                                    ;mov    [bx+BUFFINF.Cache_ptr],cx  ; [bx+6]
00000FE0 894F06                     mov     [bx+6],cx
00000FE3 2E8B0E[6403]               mov     cx,[cs:memhi]
                                    ;mov    [bx+BUFFINF.Cache_ptr+2],cx ; [bx+8]
00000FE8 894F08                     mov     [bx+8],cx
00000FEB 2E8B0E[9B02]               mov     cx,[cs:h_buffers]
                                    ;mov    [bx+BUFFINF.Cache_count],cx ; [bx+10]
00000FF0 894F0A                     mov     [bx+10],cx
00000FF3 B80002                     mov     ax,512                ; 512 byte
00000FF6 F7E1                       mul     cx
00000FF8 2EA3[6203]                 mov     [cs:memlo],ax
                                    ;or     byte [cs:setdevmarkflag],for_devmark ; 2
00000FFC 2E800E[6919]02             or      byte [cs:setdevmarkflag],2
00001002 E8ED37                     call    round
                                xif16:

                                ; ---------------------------------------------------------------------------
                                ; allocate cdss
                                ; ---------------------------------------------------------------------------

                                buf1:
00001005 E8EA37                     call    round

00001008 50                         push    ax
                                    ; 23/10/2022
                                    ;mov    ax,devmark_cds        ;='L'
00001009 B84C00                     mov     ax, 'L'
0000100C E87C06                     call    setdevmark
0000100F 58                         pop     ax

00001010 2EC43E[6D02]               les     di,[cs:DOSINFO]
                                    ;mov    cl,[es:di+SYSI_NUMIO]
00001015 268A4D20                   mov     cl,[es:di+20h]
00001019 2E3A0E[A202]               cmp     cl,[cs:NUM_CDS]
0000101E 7305                       jae     short gotncds         ; user setting must be at least numio
00001020 2E8A0E[A202]               mov     cl,[cs:NUM_CDS]
                                gotncds:
00001025 30ED                       xor     ch,ch
                                    ;mov    [es:di+SYSI_NCDS],cl  ; [es:di+33]
00001027 26884D21                   mov     [es:di+21h],cl
0000102B 2EA1[6403]                 mov     ax,[cs:memhi]
                                    ;mov    [es:di+SYSI_CDS+2],ax
0000102F 26894518                   mov     [es:di+18h],ax
00001033 2EA1[6203]                 mov     ax,[cs:memlo]
                                    ;mov    [es:di+SYSI_CDS],ax
00001037 26894516                   mov     [es:di+16h],ax
0000103B 88C8                       mov     al,cl
                                    ;mov    ah,curdirlen ; curdir_list.size
0000103D B458                       mov     ah,88
0000103F F6E4                       mul     ah
00001041 E8D102                     call    ParaRound
00001044 2E0106[6403]               add     [cs:memhi],ax

                                    ;or     byte [cs:setdevmarkflag],for_devmark ; 2
00001049 2E800E[6919]02             or      byte [cs:setdevmarkflag],2
0000104F E8A037                     call    round                 ; check for mem error before initializing
                                    ;lds    si,[es:di+SYSI_DPB] ; [es:di+0]
00001052 26C535                     lds     si,[es:di]
                                    ;les    di,[es:di+SYSI_CDS] ; [es:di+22]
00001055 26C47D16                   les     di,[es:di+16h]
00001059 E875FD                     call    fooset

                                ; ---------------------------------------------------------------------------
                                ; allocate space for internal stack
                                ; ---------------------------------------------------------------------------

0000105C 0E                         push    cs
0000105D 1F                         pop     ds

                                ;   if the user did not entered stacks= command, as a default, do not install
                                ;   sytem stacks for pc1,pc xt,pc portable cases.
                                ;   otherwise,install it to the user specified value or to the default
                                ;   value of 9,128 for other systems.

0000105E 833E[9002]FF               cmp     word [stack_addr],-1 ; has the user entered "stacks=" command?
00001063 740E                       je      short doinstallstack  ; then install as specified by the user
00001065 803E[BC02]00               cmp     byte [sys_scnd_model_byte],0 ; pc1,xt has the secondary model byte = 0
0000106A 7507                       jne     short doinstallstack  ; other model should have default stack of 9,128
0000106C 803E[BB02]FE               cmp     byte [sys_model_byte],0FEh ; pc1, pc/xt or pc portable ?
00001071 736D                       jae     short skipstack
                                doinstallstack:
00001073 A1[8C02]                   mov     ax,[stack_count]      ; stack_count = 0?
```

```
27385 00001076 09C0                     or      ax,ax                   ; then, stack size must be 0 too.
27386 00001078 7466                     jz      short skipstack                 ; don't install stack.
27387
27388                                 ;   dynamic relocation of stack code.
27389
27390 0000107A E87537                   call    round                   ;[memhi] = seg. for stack code
27391                                                                   ;[memlo] = 0
27392
27393                                 ; set devmark block into memory for mem command
27394                                 ; devmark_id = 's' for stack
27395
27396                                     ;mov    al,devmark_stk ;='S'
27397                                     ; 23/10/2022
27398 0000107D B053                     mov     al,'S'
27399 0000107F E80906                   call    setdevmark
27400
27401 00001082 A1[6403]                 mov     ax,[memhi]
27402 00001085 8EC0                     mov     es,ax           ;es -> seg. the stack code is going to move.
27403                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
27404                                   ; 11/12/2022
27405                                   ; ds = cs
27406                                   ;push   cs
27407                                   ;pop    ds
27408 00001087 31F6                     xor     si,si           ;!!we know that stack code is at the beginning of sysinit.
27409 00001089 31FF                     xor     di,di
27410 0000108B B9[6902]                 mov     cx,endstackcode
27411 0000108E 890E[6203]               mov     [memlo],cx
27412 00001092 E85D37                   call    round           ;have enough space for relocation?
27413 00001095 F3A4                     rep     movsb
27414
27415 00001097 1E                       push    ds              ; stick the location of the NextStack entry
27416                                   ;;mov   ax,Bios_Data    ; into the Win386 Instance Data tables
27417                                   ;mov    ax,KERNEL_SEGMENT ; 70h
27418                                   ; 21/10/2022
27419 00001098 B87000                   mov     ax,DOSBIODATASEG ; 0070h
27420 0000109B 8ED8                     mov     ds,ax
27421 0000109D C706[0208][1000]         mov     word [NextStack],nextentry ; (8C0h for MSDOS 6.21 IO.SYS)
27422 000010A3 8C06[0408]               mov     [NextStack+2],es          ; (8C2h for MSDOS 6.21 IO.SYS)
27423
27424 000010A7 2EA1[6203]               mov     ax,[cs:memlo]
27425 000010AB 2EA3[9002]               mov     [cs:stack_addr],ax ;set for stack area initialization
27426 000010AF A3[0808]                 mov     [IT_StackLoc],ax ; pass it as Instance Data, too
27427 000010B2 2EA1[6403]               mov     ax,[cs:memhi]  ;this will be used by stack_init routine.
27428 000010B6 2EA3[9202]               mov     [cs:stack_addr+2],ax
27429 000010BA A3[0A08]                 mov     [IT_StackLoc+2],ax
27430
27431                                 ;   space for internal stack area = stack_count(entrysize + stack_size)
27432
27433                                     ;mov    ax,entrysize ; mov ax,8
27434                                     ; 23/10/2022
27435 000010BD B80800                   mov     ax,8
27436 000010C0 2E0306[8E02]             add     ax,[cs:stack_size]
27437 000010C5 2EF726[8C02]             mul     word [cs:stack_count]
27438
27439 000010CA A3[0C08]                 mov     [IT_StackSize],ax ; pass through to Instance Tables
27440
27441 000010CD 1F                       pop     ds              ; no more need to access Instance Table
27442
27443 000010CE E84402                   call    ParaRound       ; convert size to paragraphs
27444
27445                                   ; 11/12/2022
27446                                   ; ds = cs
27447                                   ;add    [cs:memhi],ax
27448 000010D1 0106[6403]               add     [memhi],ax
27449                                   ;or     byte [cs:setdevmarkflag],for_devmark ; 2
27450                                   ;or     byte [cs:setdevmarkflag],2
27451 000010D5 800E[6919]02             or      byte [setdevmarkflag],2
27452                                   ;or     byte [setdevmarkflag],for_devmark ; 2
27453                                                           ;to set the devmark_size for stack by round routine.
27454 000010DA E81537                   call    round           ; check for memory error before
27455                                                           ; continuing
27456 000010DD E87D03                   call    stackinit       ; initialize hardware stack.
27457                                                           ; cs=ds=sysinitseg,es=relocated stack code & data
27458                                 skipstack:
27459
27460                                     ; 10/04/2024 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
27461                                     ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:11F0h)
27462                                     ;;;
27463                                     ;push   cs
27464                                     ;pop    ds
27465                                     ; ds = cs
27466 000010E0 803E[6E03]01             cmp     byte [dosdata_umb],1  ; PCDOS 7 feature - DOSDATA=UMB/NOUMB configuration
27467                                                                 ; 1 = DOSDATA=UMB, 2 = (UMB) done, 0 = NOUMB
27468 000010E5 7773                     ja      short dosdata_umb_done; 2 - done
27469 000010E7 727D                     jb      short dosdata_noumb   ; 0 - DOSDATA=NOUMB
27470
27471 000010E9 803E[8B16]EA             cmp     byte [setdevmark],0EAh
27472 000010EE 7476                     je      short dosdata_noumb
27473
27474 000010F0 B80258                   mov     ax,5802h
27475 000010F3 CD21                     int     21h             ; DOS - 3+ - GET/SET MEMORY ALLOCATION STRATEGY
27476                                                           ; AL = function code: (DOS 5beta) get UMB link state
27477 000010F5 98                       cbw
27478 000010F6 89C7                     mov     di,ax                   ; al = 01h -> UMBs in DOS memory chain
27479                                                                   ; save current (previous) UMB link state
27480 000010F8 BB0100                   mov     bx,1                    ; bx = 01h -> add UMBs to DOS memory chain
27481
27482 000010FB B80358                   mov     ax,5803h
27483 000010FE CD21                     int     21h
27484 00001100 7264                     jc      short dosdata_noumb
27485
27486 00001102 B80058                   mov     ax,5800h
27487 00001105 CD21                     int     21h             ; DOS - 3+ - GET/SET MEMORY ALLOCATION STRATEGY
27488                                                           ; AL = function code: get allocation strategy
27489
27490 00001107 89C6                     mov     si,ax                   ; ax = current strategy
27491                                                                   ; save current (previous) allocation strategy
27492 00001109 BB4000                   mov     bx,40h                  ; bl = new strategy = 40h - high memory first fit
27493
27494 0000110C B80158                   mov     ax,5801h
27495 0000110F CD21                     int     21h
27496
27497 00001111 8B1E[6403]               mov     bx,[memhi]
27498 00001115 2B1E[6A03]               sub     bx,[prev_memhi]
27499
27500 00001119 B448                     mov     ah,48h
27501 0000111B CD21                     int     21h             ; DOS - 2+ - ALLOCATE MEMORY
27502                                                           ; BX = number of 16-byte paragraphs desired
27503 0000111D 89C1                     mov     cx,ax                   ; ax = segment of allocated block
27504 0000111F 89FB                     mov     bx,di                   ; restore previous UMB link state
27505
27506 00001121 B80358                   mov     ax,5803h
27507 00001124 CD21                     int     21h             ; DOS - 3+ - GET/SET MEMORY ALLOCATION STRATEGY
27508                                                           ; AL = function code: (DOS 5beta) set UMB link state
```

```
00001126 89F3                   mov     bx,si                   ; restore previous allocation strategy

00001128 B80158                 mov     ax,5801h
0000112B CD21                   int     21h             ; DOS - 3+ - GET/SET MEMORY ALLOCATION STRATEGY
                                                        ; AL = function code: set allocation strategy
0000112D 81F900A0               cmp     cx,0A000h               ; Is the allocated memory block (segment) a UMB?
00001131 7233                   jb      short dosdata_noumb     ; no

                                ;mov    word [ALLOCLIM],0FFFFh
                                ;mov    word [memlo],0
00001133 890E[6403]             mov     [memhi],cx
00001137 49                     dec     cx
00001138 8EC1                   mov     es,cx                   ; point to arena/mcb
                                ; 10/04/2024
0000113A 31C9                   xor     cx,cx ; 0
0000113C 890E[6203]             mov     [memlo],cx ; 0
00001140 49                     dec     cx
00001141 890E[A502]             mov     [ALLOCLIM],cx ; 0FFFFh

00001145 26C70601000800         mov     word [es:1],8           ; [es:arena_owner], 8 ; set impossible owner
0000114C 26C70608005344         mov     word [es:8],4453h       ; [es:arena_name],'SD' ; System Data
00001153 FE06[6E03]             inc     byte [dosdata_umb]      ; 1 -> 2 ; DOSDATA=UMB done.
00001157 E909FD                 jmp     dosfts

                        dosdata_umb_done:
0000115A A1[6A03]               mov     ax,[prev_memhi]                 ; (recent memory block/segment before UMBs)
0000115D A3[6403]               mov     [memhi],ax
00001160 A1[6C03]               mov     ax,[prev_alloclim]
00001163 A3[A502]               mov     [ALLOCLIM],ax
                        dosdata_noumb:
                                ;;;

                        ;skipstack:
                                ; 24/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ; (SYSINIT:0F99h)

                                ; 11/12/2022
                                ; ds = cs
                                ;push   cs
                                ;pop    ds

00001166 A0[9F02]               mov     al,[FILES]
00001169 30E4                   xor     ah,ah           ; do not use cbw instruction!!!!!
                                                        ;  it does sign extend.
0000116B 89C1                   mov     cx,ax
0000116D 31DB                   xor     bx,bx           ;close standard input
0000116F B43E                   mov     ah,3Eh ; CLOSE
00001171 CD21                   int     21h
00001173 BB0200                 mov     bx,2
                        rcclloop:                       ;close everybody but standard output
00001176 B43E                   mov     ah,3Eh ; CLOSE ; need output so we can print message
00001178 CD21                   int     21h             ; in case we can't get new one open.
0000117A 43                     inc     bx
0000117B E2F9                   loop    rcclloop

0000117D BA[C54A]               mov     dx,condev
00001180 B002                   mov     al,2
00001182 B43D                   mov     ah,3Dh ; OPEN  ;open con for read/write
00001184 F9                     stc                     ; set for possible int 24
00001185 CD21                   int     21h
00001187 7305                   jnc     short goaux
00001189 E89C38                 call    badfil
0000118C EB13                   jmp     short goaux2
                        goaux:
0000118E 50                     push    ax
0000118F BB0100                 mov     bx,1            ;close standard output
00001192 B43E                   mov     ah,3Eh ; CLOSE
00001194 CD21                   int     21h
00001196 58                     pop     ax

00001197 89C3                   mov     bx,ax           ;new device handle
00001199 B445                   mov     ah,45h ; XDUP
0000119B CD21                   int     21h             ;dup to 1,stdout
0000119D B445                   mov     ah,45h ; XDUP
0000119F CD21                   int     21h             ;dup to 2,stderr
                        goaux2:
000011A1 BA[C94A]               mov     dx,auxdev
000011A4 B002                   mov     al,2            ;read/write access
000011A6 E8B038                 call    open_dev

000011A9 BA[CD4A]               mov     dx,prndev
000011AC B001                   mov     al,1            ;write only
000011AE E8A838                 call    open_dev

                        ;global rearm command for shared interrupt devices attached in the system;
                        ;shared interrupt attachment has some problem when it issues interrupt
                        ;during a warm reboot. once the interrupt is presented by the attachment,
                        ;no further interrupts on that level will be presented until a global rearm
                        ;is issued. by the request of the system architecture group, msbio will
                        ;issue a global rearm after every device driver is loaded.
                        ;to issue a global rearm: ;for pc1,xt,palace
                        ;
                        ;                       out 02f2h,xx  ; interrupt level 2
                        ;                       out 02f3h,xx  ; interrupt level 3
                        ;                       out 02f4h,xx  ; interrupt level 4
                        ;                       out 02f5h,xx  ; interrupt level 5
                        ;                       out 02f6h,xx  ; interrupt level 6
                        ;                       out 02f7h,xx  ; interrupt level 7
                        ;
                        ;   for pc at,in addition to the above commands,
                        ;   need to handle the secondary interrupt handler
                        ;
                        ;                       out 06f2h,xx  ; interrupt level 10
                        ;                       out 06f3h,xx  ; interrupt level 11
                        ;                       out 06f4h,xx  ; interrupt level 12
                        ;                       out 06f6h,xx  ; interrupt level 14
                        ;                       out 06f7h,xx  ; interrupt level 15
                        ;
                        ;   for round-up machine
                        ;
                        ;                       none.

                        ; where xx stands for any value.
                        ;
                        ; for your information,after naples level machine,the system service bios
                        ; call (int 15h),function ah=0c0h returns the system configuration parameters

                            ; 24/10/2022

000011B1 50                 push    ax
000011B2 53                 push    bx
000011B3 52                 push    dx
000011B4 06                 push    es
```

```
27633 000011B5 B0FF                      mov     al,0FFh             ;reset h/w by writing to port
27634 000011B7 BAF202                    mov     dx,2F2h             ;get starting address
27635 000011BA EE                        out     dx,al               ; out 02f2h,0ffh
27636 000011BB 42                        inc     dx
27637 000011BC EE                        out     dx,al               ; out 02f3h,0ffh
27638 000011BD 42                        inc     dx
27639 000011BE EE                        out     dx,al               ; out 02f4h,0ffh
27640 000011BF 42                        inc     dx
27641 000011C0 EE                        out     dx,al               ; out 02f5h,0ffh
27642 000011C1 42                        inc     dx
27643 000011C2 EE                        out     dx,al               ; out 02f6h,0ffh
27644 000011C3 42                        inc     dx
27645 000011C4 EE                        out     dx,al               ; out 02f7h,0ffh
27646
27647                                 ;sb secondary global rearm
27648
27649 000011C5 B800F0                    mov     ax,0F000h           ;get machine type
27650 000011C8 8EC0                      mov     es,ax
27651 000011CA 26803EFEFFFC              cmp     byte [es:0FFFEh],0FCh ;q:is it a at type machine
27652 000011D0 740D                      je      short startrearm    ; *if at no need to check
27653
27654 000011D2 B4C0                      mov     ah,0C0h             ;get system configuration
27655 000011D4 CD15                      int     15h                 ; *
27656 000011D6 7216                      jc      short finishrearm   ; *jmp if old rom
27657
27658                                 ; test feature byte for secondary interrupt controller
27659
27660 000011D8 26F6470540                test    byte [es:bx+5],40h
27661                                    ; 24/10/2022
27662                                    ;test   byte [es:bx+ROMBIOS_DESC.bios_sd_featurebyte1],ScndIntController
27663 000011DD 740F                      je      short finishrearm   ;jmp if it is there
27664
27665                                 startrearm:
27666 000011DF B0FF                      mov     al,0FFh             ;write any pattern to port
27667 000011E1 BAF206                    mov     dx,6F2h             ;get starting address
27668 000011E4 EE                        out     dx,al               ;out 06f2h,0ffh
27669 000011E5 42                        inc     dx                  ;bump address
27670 000011E6 EE                        out     dx,al               ;out 06f3h,0ffh
27671 000011E7 42                        inc     dx                  ;bump address
27672 000011E8 EE                        out     dx,al               ;out 06f4h,0ffh
27673 000011E9 42                        inc     dx                  ;bump address
27674 000011EA 42                        inc     dx                  ;bump address
27675 000011EB EE                        out     dx,al               ;out 06f6h,0ffh
27676 000011EC 42                        inc     dx                  ;bump address
27677 000011ED EE                        out     dx,al               ;out 06f7h,0ffh
27678
27679                                 finishrearm:
27680 000011EE 07                        pop     es
27681 000011EF 5A                        pop     dx
27682 000011F0 5B                        pop     bx
27683 000011F1 58                        pop     ax
27684
27685                                 ;    global rearm end ********************
27686
27687                                 ; ---------------------------------------------------------------------
27688                                 ; allocate sysinit_base for install= command
27689                                 ; ---------------------------------------------------------------------
27690                                 ; sysinit_base allocation.
27691                                 ;   check if endfile has been called to handle install= command.
27692
27693                                 set_sysinit_base:
27694
27695                                 ; ---------------------------------------------------------------------
27696                                 ;sysinit_base will be established in the secure area of
27697                                 ;lower memory when it handles the first install= command.
27698                                 ;sysinit_base is the place where the actual exec function will be called and
27699                                 ;will check sysinit module in high memory if it is damaged by the application
27700                                 ;program.  if sysinit module has been broken,then "memory error..." message
27701                                 ;is displayed by sysinit_base.
27702                                 ; ---------------------------------------------------------------------
27703
27704                                     ; 24/10/2022 - Retro DOS v4.0 (MSDOS 5.0 IO.SYS, SYSINIT)
27705                                     ; (SYSINIT:1028h)
27706
27707                                     ; 11/12/2022
27708                                     ; ds = cs
27709 000011F2 50                        push    ax                  ; set devmark for mem command
27710 000011F3 A1[6403]                  mov     ax,[memhi]
27711 000011F6 2B06[6803]                sub     ax,[area]
27712 000011FA A3[6003]                  mov     [impossible_owner_size],ax ;remember the size in case.
27713                                    ;mov    al,devmark_inst ; 'T'
27714 000011FD B054                      mov     al,'T'
27715 000011FF E88904                    call    setdevmark
27716 00001202 58                        pop     ax
27717
27718 00001203 8B3E[6403]                mov     di,[memhi]
27719 00001207 8EC7                      mov     es,di
27720 00001209 893E[D402]                mov     [sysinit_base_ptr+2],di ; save this entry for the next use.
27721 0000120D 31FF                      xor     di,di
27722 0000120F 893E[D202]                mov     [sysinit_base_ptr],di ; es:di -> destination.
27723 00001213 BE[2113]                  mov     si,sysinit_base             ;ds:si -> source code to be relocated.
27724 00001216 B98100                    mov     cx,end_sysinit_base-sysinit_base ; 129
27725                                    ; 24/10/2022
27726                                    ;mov    cx,128  ; 11DCh-115Ch  ; (MSDOS 5.0 IO.SYS, SYSINIT)
27727 00001219 010E[6203]                add     [memlo],cx
27728                                    ;or     byte cs:[setdevmarkflag],for_devmark ; 2
27729                                    ; 11/12/2022
27730                                    ; ds = cs
27731                                    ;or     byte [cs:setdevmarkflag],2
27732 0000121D 800E[6919]02              or      byte [setdevmarkflag],2
27733                                    ;or     byte [setdevmarkflag],for_devmark
27734 00001222 E8CD35                    call    round               ; check mem error. also,readjust memhi for the next use.
27735 00001225 F3A4                      rep     movsb               ; reallocate it.
27736
27737 00001227 C706[D602][0813]           mov     word [sysinit_ptr],sysinitptr ; returning address from
27738 0000122D 8C0E[D802]                mov     [sysinit_ptr+2],cs      ; sysinit_base back to sysinit.
27739                                    ;or     word [install_flag],has_installed ; set the flag.
27740                                    ;or     byte [install_flag],has_installed ; 2
27741                                    ; 11/12/2022
27742 00001231 800E[CE02]02              or      byte [install_flag],2
27743                                    ; 24/10/2022
27744                                    ;or     word [install_flag],2
27745
27746                                 ; ---------------------------------------------------------------------
27747                                 ; free the rest of the memory from memhi to confbot. still from confbot to
27748                                 ; the top of the memory will be allocated for sysinit and config.sys if
27749                                 ; have_install_cmd.
27750                                 ; ---------------------------------------------------------------------
27751
27752 00001236 E8B935                    call    round
27753 00001239 8B1E[6403]                mov     bx,[memhi]
27754 0000123D A1[6803]                  mov     ax,[area]
27755 00001240 A3[5E03]                  mov     [old_area],ax       ; save [area]
27756 00001243 8EC0                      mov     es,ax               ;calc what we needed
```

```
00001245 29C3                           sub     bx,ax
                                        ; 24/10/2022
00001247 B44A                           mov     ah,4Ah ; SETBLOCK
00001249 CD21                           int     21h                     ;give the rest back
                                                ; DOS - 2+ - ADJUST MEMORY BLOCK SIZE (SETBLOCK)
                                                ; ES = segment address of block to change
                                                ; BX = new size in paragraphs
0000124B 06                             push    es
0000124C 8CC0                           mov     ax,es
0000124E 48                             dec     ax
0000124F 8EC0                           mov     es,ax                   ;point to arena
                                        ;mov    word [es:ARENA.OWNER],8       ;set impossible owner
                                        ;;mov   word [es:ARENA.NAME],4453h    ; System Data
                                        ;mov    word [es:ARENA.NAME],'SD'     ; System Data
                                        ; 24/10/2022
00001251 26C70601000800                 mov     word [es:1],8           ;set impossible owner
00001258 26C70608005344                 mov     word [es:8],'SD'        ; System Data
0000125F 07                             pop     es

00001260 BBFFFF                         mov     bx,0FFFFh
00001263 B448                           mov     ah,48h ; ALLOC
00001265 CD21                           int     21h
00001267 B448                           mov     ah,48h ; ALLOC
00001269 CD21                           int     21h                     ; allocate the rest of the memory
                                                ; DOS - 2+ - ALLOCATE MEMORY
                                                ; BX = number of 16-byte paragraphs desired
0000126B A3[6403]                       mov     [memhi],ax              ; start of the allocated memory
0000126E C706[6203]0000                 mov     word [memlo],0          ;  to be used next.

                                ;;;; at this moment,memory from [memhi]:0 to top-of-the memory is
                                ;;;; allocated.
                                ;;;; to protect sysinit,confbot module (from confbot (or =alloclim at
                                ;;;; this time) to the top-of-the memory),here we are going to
                                ;;;; 1). "setblock" from memhi to confbot.
                                ;;;; 2). "alloc" from confbot to the top of the memory.
                                ;;;; 3). "free alloc memory" from memhi to confbot.

                                ;memory allocation for sysinit,confbot module.

00001274 8EC0                           mov     es,ax
                                        ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                        ; (SYSINIT:11DFh)
00001276 8B1E[A302]                     mov     bx,[CONFBOT]
                                        ; 24/10/2022
                                        ;mov    bx,[top_of_cdss] ; mov bx,[confbot]
0000127A 29C3                           sub     bx,ax                   ; confbot - memhi
0000127C 4B                             dec     bx                      ; make a room for the memory block id.
0000127D 4B                             dec     bx                      ; make sure!!!.
0000127E B44A                           mov     ah,4Ah ; SETBLOCK
00001280 CD21                           int     21h                     ; this will free (confbot to top of memory)
                                                ; DOS - 2+ - ADJUST MEMORY BLOCK SIZE (SETBLOCK)
                                                ; ES = segment address of block to change
                                                ; BX = new size in paragraphs
00001282 BBFFFF                         mov     bx,0FFFFh
00001285 B448                           mov     ah,48h ; ALLOC
00001287 CD21                           int     21h
00001289 B448                           mov     ah,48h ; ALLOC
0000128B CD21                           int     21h                     ; allocate (confbot to top of memory)
                                                ; DOS - 2+ - ALLOCATE MEMORY
                                                ; BX = number of 16-byte paragraphs desired
0000128D A3[6803]                       mov     [area],ax               ; save allocated memory segment.
                                                                        ; need this to free this area for command.com.
00001290 8E06[6403]                     mov     es,[memhi]
00001294 B449                           mov     ah,49h                  ; free allocated memory.
00001296 CD21                           int     21h                     ; free (memhi to confbot(=area))
                                                ; DOS - 2+ - FREE MEMORY
                                                ; ES = segment address of area to be freed
                                endfile_ret:
00001298 C3                             retn

                                ; End of "EndFile" DOS structure configuration.

                                ; ---------------------------------------------------------------------------
                                ; 26/03/2019 - Retro DOS v4.0
                                ; 24/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS, SYSINIT)
                                ; ---------------------------------------------------------------------------
                                ; Do_Install_Exec
                                ;
                                ; This procedure is used to EXEC a program being loaded via the
                                ; "install=" mechanism in config.sys. It does this by setting up
                                ; the parameters, and then jumping to sysinit_base, which has been
                                ; setup in low memory. When complete, sysinit_base will jump back
                                ; up to this procedure (if sysinit remains uncorrupted by the installed
                                ; program).

                                ;SYSINIT:10CFh:

                                do_install_exec:                        ; now,handles install= command.

00001299 56                             push    si                      ; save si for config.sys again.

                                ; we are going to call load/exec function.
                                ; set es:bx to the parameter block here;;;;;;;
                                ; set ds:dx to the asciiz string. remember that we already has 0
                                ; after the filename. so parameter starts after that. if next
                                ; character is a line feed (i.e. 10),then assume that the 0
                                ; we already encountered used to be a carrage return. in this
                                ; case,let's set the length to 0 which will be followed by
                                ; carridge return.

                                ; es:si -> command line in config.sys. points to the first non blank
                                ;character after =.

0000129A 06                             push    es
0000129B 1E                             push    ds
0000129C 07                             pop     es
0000129D 1F                             pop     ds                      ; es->sysinitseg,ds->confbot seg
0000129E 89F2                           mov     dx,si                   ; ds:dx->file name,0 in config.sys image.

000012A0 31C9                           xor     cx,cx
000012A2 FC                             cld
000012A3 2EC606[F102]20                 mov     byte [cs:ldexec_start],' ' ; clear out the parm area
000012A9 BF[F202]                       mov     di,ldexec_parm
                                installfilename:                        ; skip the file name
000012AC AC                             lodsb                           ; al = ds:si; si++
                                        ; 05/09/2023
000012AD 08C0                           or      al,al
                                        ;cmp    al,0
                                        ;je     short got_installparm
                                        ;jmp    short installfilename
                                        ; 10/04/2024
000012AF 75FB                           jnz     short installfilename
                                got_installparm:                        ; copy the parameters to ldexec_parm
000012B1 AC                             lodsb
```

```
27881 000012B2 268805                   mov     [es:di],al
27882 000012B5 3C0A                     cmp     al,lf   ; cmp al,0Ah   ; line feed?
27883 000012B7 7405                     je      short done_installparm
27884 000012B9 FEC1                     inc     cl                      ; # of char. in the parm.
27885 000012BB 47                       inc     di
27886 000012BC EBF3                     jmp     short got_installparm
27887                                 done_installparm:
27888 000012BE 2E880E[F002]             mov     byte [cs:ldexec_line],cl ; length of the parm.
27889                                   ; 05/09/2023
27890 000012C3 08C9                     or      cl,cl
27891                                   ;cmp    cl,0                    ; if no parm,then
27892 000012C5 7506                     jne     short install_seg_set ; let the parm area
27893 000012C7 2EC606[F102]0D           mov     byte [cs:ldexec_start],cr ; 0Dh
27894                                                                   ; starts with cr.
27895                                 install_seg_set:
27896                                   ; 05/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
27897 000012CD 31DB                     xor     bx,bx
27898                                   ;mov    word [cs:0],0           ; make a null environment segment
27899 000012CF 2E891F                   mov     [cs:bx],bx ; 05/09/2023
27900 000012D2 8CC8                     mov     ax,cs                   ; by overlap jmp instruction of sysinitseg.
27901
27902                                 ;--------------------------------------------------M067----------------
27903                                 ;
27904                                 ;   the environment pointer is made 0. so the current environment ptr.
27905                                 ;   will be the same as pdb_environ which after dosinit is 0.
27906                                 ;
27907                                 ;   mov     cs:[instexe.exec0_environ],0 ; set the environment seg.
27908                                 ;
27909                                 ;   instexe.exec0_environ need not be initialized to 0 above. It was
27910                                 ;   done as a fix for bug #529. The actual bug was in NLSFUNC and
27911                                 ;   was fixed.
27912                                 ;
27913                                 ; ------------------------------------------------------------------------
27914
27915                                 ;;ifdef MULTI_CONFIG
27916
27917                                 ; If there's any environment data in "config_wrkseg", pass to app
27918                                 ; 30/12/2022 - Retro DOS v4.0 (Modified MSDOS 6.21 IO.SYS SYSINIT)
27919                                 ; 24/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS SYSINIT)
27920                                 ;%if 0
27921 000012D4 89C1                     mov     cx,ax ; *
27922                                   ; 05/09/2023
27923 000012D6 2E391E[6019]             cmp     [cs:config_envlen],bx ; 0
27924                                   ;cmp    word [cs:config_envlen],0
27925 000012DB 7405                     je      short no_envdata2
27926 000012DD 2E8B0E[6219]             mov     cx,[cs:config_wrkseg] ; *
27927                                 no_envdata2:
27928                                 ;;endif  ;MULTI_CONFIG
27929
27930                                 ;%endif    ; 24/10/2022
27931
27932                                     ;mov    [cs:instexe.exec0_environ],cx ; set the environment seg.
27933                                     ; 05/09/2023 (BugFix)
27934                                     ; 24/10/2022
27935 000012E2 2E890E[4203]             mov     [cs:iexec.environ],cx ; *
27936                                     ; 02/11/2022
27937                                     ;mov    [cs:iexec.environ],ax ; 05/09/2023
27938
27939                                     ;mov    [cs:instexe.exec0_com_line+2],ax ; set the seg.
27940 000012E7 2EA3[4603]                mov     [cs:iexec.ldexec_line+2],ax
27941                                     ;mov    [cs:instexe.exec0_5c_fcb+2],ax
27942 000012EB 2EA3[4A03]                mov     [cs:iexec.ldexec_5c_fcb+2],ax
27943                                     ;mov    [cs:instexe.exec0_6c_fcb+2],ax
27944 000012EF 2EA3[4E03]                mov     [cs:iexec.ldexec_6c_fcb+2],ax
27945 000012F3 E86000                    call    sum_up
27946 000012F6 26A3[DA02]                mov     [es:checksum],ax        ; save the value of the sum
27947 000012FA 31C0                      xor     ax,ax
27948 000012FC B44B                      mov     ah,4Bh ; EXEC           ; load/exec
27949 000012FE BB[4203]                  mov     bx,instexe              ; es:bx -> parm block.
27950 00001301 06                        push    es                      ; save es,ds for load/exec
27951 00001302 1E                        push    ds                      ; these registers will be restored in sysinit_base.
27952 00001303 2EFF2E[D202]              jmp     far [cs:sysinit_base_ptr] ; jmp to sysinit_base to execute
27953                                                                    ; load/exec function and check sum.
27954
27955                                 ;---------------------------------------
27956
27957                                 ;j.k. this is the returning address from sysinit_base.
27958
27959                                     ; 24/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS SYSINIT)
27960
27961                                 sysinitptr:                         ; returning far address from sysinit_base
27962 00001308 5E                        pop     si                      ; restore si for config.sys file.
27963 00001309 06                        push    es
27964 0000130A 1E                        push    ds
27965 0000130B 07                        pop     es
27966 0000130C 1F                        pop     ds                      ; now ds - sysinitseg, es - confbot
27967 0000130D 7305                          jnc     short install_exit_ret
27968
27969 0000130F 56                        push    si                      ; error in loading the file for install=.
27970 00001310 E81937                    call    badload                 ; es:si-> path,filename,0.
27971 00001313 5E                        pop     si
27972
27973                                     ; 24/10/2022
27974                                     ;jmp    short sysinitptr_retn ; (MSDOS 5.0 IO.SYS, SYSINIT:1140h)
27975                                     ; 11/12/2022
27976                                     ; ds = cs
27977
27978                                     ; 30/12/2022 - Retro DOS v4.2
27979                                     ; (MSDOS 6.21 IO.SYS, SYSINIT:1283h)
27980
27981                                 install_exit_ret:
27982 00001314 C3                        retn
27983
27984                                 ; 30/12/2022 - Retro DOS v4.2
27985                                 %if 0
27986                                 install_exit_ret:
27987                                     ;retn          ; retn (MSDOS 6.21 IO.SYS, SYSINIT:1283h) ; 18/12/2022
27988
27989                                     ; 24/10/2022 (MSDOS 5.0 IO.SYS SYSINIT)
27990                                 ;SYSINIT:1142h:
27991                                     mov     ah,4Dh
27992                                     int     21h             ; DOS - 2+ - GET EXIT CODE OF SUBPROGRAM (WAIT)
27993                                     cmp     ah,3
27994                                     jz      short sysinitptr_retn
27995                                     call    error_line
27996                                     stc
27997                                 sysinitptr_retn:   ; (SYSINIT:114Fh)
27998                                     retn
27999
28000                                 %endif ; 24/10/2022
28001
28002                                 ; ------------------------------------------------------------------------
28003
28004
```

```
                                ;** ParaRound - Round Up length to paragraph multiple
                                ;
                                ;   ParaRound rounds a byte count up to a multiple of 16, then divides
                                ;   by 16 yielding a "length in paragraphs" value.
                                ;
                                ;   ENTRY   (ax) = byte length
                                ;   EXIT    (ax) = rounded up length in paragraphs
                                ;   USES    ax, flags

                                ParaRound:
00001315 83C00F                     add     ax,15
00001318 D1D8                       rcr     ax,1
0000131A D1E8                       shr     ax,1
0000131C D1E8                       shr     ax,1
0000131E D1E8                       shr     ax,1
00001320 C3                         retn

                                ; ----------------------------------------------------------------------
                                ; sysinit_base module.
                                ;
                                ; This module is relocated by the routine EndFile to a location in low
                                ; memory. It is then called by SYSINIT to perform the EXEC of programs
                                ; that are being loaded by the "install=" command. After the EXEC call
                                ; completes, this module performs a checksum on the SYSINIT code (at the
                                ; top of memory) to be sure that the EXECed program did not damage it.
                                ; If it did, then this module will print an error message and stop the
                                ; system. Otherwise, it returns control to SYSINIT.
                                ;
                                ;in: after relocation,
                                ;    ax = 4b00h - load and execute the program dos function.
                                ;    ds = confbot. segment of config.sys file image
                                ;    es = sysinitseg. segment of sysinit module itself.
                                ;    ds:dx = pointer to asciiz string of the path,filename to be executed.
                                ;    es:bx = pointer to a parameter block for load.
                                ;    SI_end (byte) - offset vaule of end of sysinit module label
                                ;    bigsize (word) - # of word from confbot to SI_end.
                                ;    chksum (word) - sum of every byte from confbot to SI_end in a
                                ;                    word boundary moduler form.
                                ;    sysinit_ptr (dword ptr) - return address to sysinit module.
                                ;
                                ;note: sysinit should save necessary registers and when the control is back

                                    ; 24/10/2022
                                    ; (SYSINIT:115Ch for MSDOS 5.0 SYSINIT)
                                sysinit_base:
00001321 2E8C166200                 mov     [cs:sysinit_base_ss],ss       ; save stack
00001326 2E89266400                 mov     [cs:sysinit_base_sp],sp
0000132B CD21                       int     21h                     ; load/exec dos call.
0000132D 2E8E166200                 mov     ss,[cs:sysinit_base_ss]       ; restore stack
00001332 2E8B266400                 mov     sp,[cs:sysinit_base_sp]
00001337 1F                         pop     ds                      ; restore confbot seg
00001338 07                         pop     es                      ; restore sysinitseg
00001339 7216                       jc      short sysinit_base_end; load/exec function failed.
                                                                    ; at this time,i don't have to worry about
                                                                    ; that sysinit module has been broken or not.
0000133B E81800                     call    sum_up                  ; otherwise,check if it is good.
0000133E 263906[DA02]               cmp     [es:checksum],ax
00001343 740C                       je      short sysinit_base_end

                                ;   memory broken. show "memory allocation error" message and stall.

00001345 B409                       mov     ah,9
00001347 0E                         push    cs
00001348 1F                         pop     ds
                                    ; 30/12/2022
                                    ; (MSDOS 6.21 IO.SYS, SYSINIT:12B8h)
                                    ;mov    dx,102
00001349 BA6600                     mov     dx,mem_alloc_err_msgx-sysinit_base ; 65h (for MSDOS 5.0 SYSINIT)
                                                                    ; 66h (for MSDOS 6.21 SYSINIT)
0000134C CD21                       int     21h
                                            ; DOS - PRINT STRING
                                            ; DS:DX -> string terminated by "$"

                                    ; 30/12/2022 - Retro DOS v4.2
                                stall:
                                    ; 24/10/2022
                                _stall:
                                    ; 11/12/2022
0000134E F4                         hlt
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;hlt                            ;use HLT to minimize energy consumption
0000134F EBFD                           jmp short _stall

                                sysinit_base_end:
00001351 26FF2E[D602]               jmp     far [es:sysinit_ptr]  ;return back to sysinit module

                                ;---------------------------------

                                sum_up:

                                ;in:   es - sysinitseg.
                                ;out:  ax - result
                                ;
                                ;remark: since this routine will only check starting from "locstack" to the end of
                                ;    sysinit segment,the data area, and the current stack area are not
                                ;    coverd. in this sense,this check sum routine only gives a minimal
                                ;    gaurantee to be safe.
                                ;
                                ;first sum up confbot seg.

00001356 1E                         push    ds
                                    ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:12C6h)
00001357 26A1[A302]                 mov     ax,[es:CONFBOT]
                                    ; 24/10/2022
                                    ;mov    ax,[es:top_of_cdss]
0000135B 8ED8                       mov     ds,ax
0000135D 31F6                       xor     si,si
0000135F 31C0                       xor     ax,ax
00001361 268B0E[D002]               mov     cx,[es:config_size]   ; if config_size has been broken,then this
                                                                  ; whole test better fail.
00001366 D1E9                       shr     cx,1                  ; make it a word count
00001368 7406                       jz      short sum_sys_code    ; when config.sys file not exist.
                                sum1:
0000136A 0304                       add     ax,[si]
0000136C 46                         inc     si
0000136D 46                         inc     si
0000136E E2FA                       loop    sum1
                                ;now,sum up sysinit module.
                                sum_sys_code:
                                    ; 24/10/2022
00001370 BE7013                     mov     si,locstack ;; 5A6h (MSDOS 5.0 IO.SYS, SYSINIT)
                                                        ;; 532h (MSDOS 6.21 IO.SYS, SYSINIT)
                                                        ; 10/04/2024
```

```
                                                        ; 586h (PCDOS 7.1 IBMBIO.COM, SYSINIT)
                                                                ; starting after the stack.  M069
                                                               ;  this does not cover the possible stack code!!!
                                    ;;;mov  cx,22688  ; for MSDOS 6.21 IO.SYS
                                    ; 02/11/2022
                                    ;;mov   cx,3D20h  ; (15648) for MSDOS 5.0 IO.SYS (SYSINIT)
                                    ; 10/04/2024
                                    ;mov    cx,5B40h  ; (23360) for PCDOS 7.1 IBMBIO.COM (SYSINIT)
                                    ; 30/12/2022
00001373 B9[1054]                   mov     cx,SI_end ; (22688)   ; SI_end is the label at the end of sysinit
00001376 29F1                       sub     cx,si                 ;  from after_checksum to SI_end
00001378 D1E9                       shr     cx,1
                                sum2:
0000137A 260304                     add     ax,[es:si]
0000137D 46                         inc     si
0000137E 46                         inc     si
0000137F E2F9                       loop    sum2
00001381 1F                         pop     ds
00001382 C3                         retn

                                ; 24/10/2022 - Retro DOS 4.0 (Modified MSDOS 5.0 IO.SYS, SYSINIT)
                                ; 30/12/2022 - Retro DOS 4.2 (Modified MSDOS 6.21 IO.SYS, SYSINIT)
                                ; (SYSINIT:12F2h)
                                ; 10/04/2024 - Retro DOS 5.0 (Modified PCDOS 7.1 IBMBIO.COM, SYSINIT)
                                ; (SYSINIT:149Dh)

                                sysinit_base_ss equ $-sysinit_base  ; = 61 (MSDOS 5.0 IO.SYS, SYSINIT:115Ch)
                                ;SYSINIT:11BDh:                     ; = 62 (MSDOS 6.21 IO.SYS, SYSINIT:1290h)
                                                                    ; = 62 (PCDOS 7.1 IBMBIO.COM, SYSINIT:143Bh)
                                sysinit_base_ssx:
00001383 0000                       dw      0
                                sysinit_base_sp equ $-sysinit_base  ; = 63 (MSDOS 5.0 IO.SYS, SYSINIT:1161h)
                                ;SYSINIT:11BFh:                     ; = 64 (MSDOS 6.21 IO.SYS, SYSINIT:1295h)
                                                                    ; = 64 (PCDOS 7.1 IBMBIO.COM, SYSINIT:1440h)
                                sysinit_base_spx:
00001385 0000                       dw      0

                                mem_alloc_err_msgx:

                                       ;include msbio.cl4        ; memory allocation error message

                                ;(SYSINIT:12F6h: ; MSDOS 6.21 IO.SYS)
                                ;SYSINIT:14A1h:    ; PCDOS 7.1 IBMBIO.COM
00001387 0D0A                       db      0Dh,0Ah
00001389 4D656D6F727920616C-        db      'Memory allocation error $'
00001392 6C6F636174696F6E20-
0000139B 6572726F722024

                                end_sysinit_base: ; label byte
                                    ; 24/10/2022
                                    ; (SYSINIT:11DCh for MSDOS 5.0 SYSINIT)

                                ; ---------------------------------------------------------------------------
                                ; Set_Buffer
                                ;
                                ;function: set buffers in the real memory.
                                ;       lastly set the memhi,memlo for the next available free address.
                                ;
                                ;input:    ds:bx -> buffinfo.
                                ;       [memhi]:[memlo = 0] = available space for the hash bucket.
                                ;       singlebuffersize = buffer header size + sector size
                                ;
                                ;output:   buffers Queue established.
                                ;       [memhi]:[memlo] = address of the next available free space.
                                ; ---------------------------------------------------------------------------

                                    ; 25/10/2022 - Retro DOS 4.0 (Modified MSDOS 5.0 IO.SYS, SYSINIT)
                                    ; (SYSINIT:11DCh)

                                    ; 11/04/2024 - Retro DOS 5.0 (Modified PCDOS 7.1 IBMBIO.COM)
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:14BCh)

                                set_buffer:
000013A2 30D2                       xor     dl,dl                       ; assume buffers not in HMA
000013A4 E85500                     call    GetBufferAddr
000013A7 7402                       jz      short set_buff_1
000013A9 B201                       mov     dl,1                        ; buffers in HMA
                                set_buff_1:
                                    ; 25/10/2022
                                    ;mov    [bx+BUFFINF.Buff_Queue],di    ; head of Buff Q
000013AB 893F                       mov     [bx],di
                                    ;mov    [bx+BUFFINF.Buff_Queue+2],es
000013AD 8C4702                     mov     [bx+2],es
                                    ;mov    word [bx+BUFFINF.Dirty_Buff_Count],0 ;set dirty_count to 0.
000013B0 C747040000                 mov     word [bx+4],0

000013B5 89F8                       mov     ax,di
000013B7 2E8B0E[9902]               mov     cx,[cs:buffers]
000013BC 57                         push    di                          ; remember first buffer

                                ;   for each buffer

                                nxt_buff:
000013BD E87500                     call    set_buffer_info             ; set buf_link,buf_id...
000013C0 89C7                       mov     di,ax
000013C2 E2F9                       loop    nxt_buff

000013C4 2E2B3E[9D02]               sub     di,[cs:singlebuffersize]    ; point to last buffer

000013C9 59                         pop     cx                          ; get first buffer
                                    ;mov    [es:di+buffinfo.buf_next],cx ; last->next = first
000013CA 26890D                     mov     [es:di],cx
000013CD 87F9                       xchg    cx,di
                                    ;mov    [es:di+buffinfo.buf_prev],cx ; first->prev = last
                                    ; 25/10/2022
000013CF 26894D02                   mov     [es:di+2],cx

000013D3 08D2                       or      dl,dl                       ; In HMa ?
000013D5 7417                       jz      short set_buff_2            ; no
                                    ;mov    byte [bx+BUFFINF.Buff_In_HMA],1
000013D7 C6470C01                   mov     byte [bx+12],1
000013DB 2EA1[6403]                 mov     ax,[cs:memhi]               ; seg of scratch buff
                                    ;mov    word [bx+BUFFINF.Lo_Mem_Buff],0      ; offset of sctarch buff is 0
000013DF C7470D0000                 mov     word [bx+13],0
                                    ;mov    [bx+BUFFINF.Lo_Mem_Buff+2],ax
000013E4 89470F                     mov     word [bx+15],ax
000013E7 2EA1[9D02]                 mov     ax,[cs:singlebuffersize]    ; size of scratch buff
                                    ; 11/04/2024 - Retro DOS v5.0
                                    ; 05/09/2023
                                    ;;sub   ax,bufinsiz ; 20            ; buffer head not required
                                    ;sub    ax,20
000013EB 83E818                     sub     ax,24 ; bufinsiz            ; (bufinsiz is 24 in PCDOS 7.1)

                                set_buff_2:
```

```
28251 000013EE 2E0106[6203]               add     [cs:memlo],ax
28252                                     ;or     byte [cs:setdevmarkflag],for_devmark ; 2
28253 000013F3 2E800E[6919]02             or      byte [cs:setdevmarkflag],2
28254                                     ;call   round
28255                                     ;retn
28256                                     ; 12/12/2022
28257 000013F9 E9F633                     jmp     round
28258
28259                                   ; ----------------------------------------------------------------------
28260                                   ; procedure : GetBufferAddr
28261                                   ;
28262                                   ;       Gets the buffer address either in HMA or in Lo Mem
28263                                   ;
28264                                   ; returns in es:di the buffer adress
28265                                   ; returns NZ if allocated in HMA
28266                                   ; ----------------------------------------------------------------------
28267
28268                                       ; 25/10/2022
28269                                   GetBufferAddr:
28270 000013FC 53                         push    bx
28271 000013FD 52                         push    dx
28272
28273                                     ; 11/04/2024 - Retro DOS v5.0
28274                                     ; PCDOS 7.1 IBMBIO.COM
28275                                     ;;;
28276 000013FE 2E803E[6E03]02             cmp     byte [cs:dosdata_umb],2
28277                                                     ; is dosdata moved to UMB ? (DOSDATA=UMB done)
28278 00001404 7506                       jne     short gba_1     ; no
28279 00001406 837F02FF                   cmp     word [bx+2],0FFFFh ; is the buffer (already) in HMA ?
28280 0000140A 7423                       je      short gba_2    ; yes
28281                                   gba_1:
28282                                     ;;;
28283
28284 0000140C 2EA1[9D02]                 mov     ax, [cs:singlebuffersize]
28285 00001410 2EF726[9902]               mul     word [cs:buffers]
28286                                     ;add    ax,0Fh
28287 00001415 83C00F                     add     ax,15
28288                                     ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
28289                                     ;and    ax,~15  ; 0FFF0h        ; para round
28290                                     ; 12/12/2022
28291 00001418 24F0                       and     al,~15  ; 0F0h
28292 0000141A 89C3                       mov     bx,ax
28293 0000141C B8024A                     mov     ax,4A02h
28294                                     ;mov    ax,((multMULT<<8)+multMULTALLOCHMA)
28295 0000141F CD2F                       int     2Fh     ; DOS 5+ - ALLOCATE HMA SPACE
28296                                                     ;     AX = 4A02h
28297                                                     ;     BX = number of bytes
28298                                                     ; Return:
28299                                                     ;     ES:DI -> start of allocated HMA block or FFFFh:FFFFh
28300                                                     ;     BX = number of bytes actually allocated
28301                                                     ;          (rounded up to next paragraph)
28302                                                     ; Notes:
28303                                                     ;     this call is not valid unless DOS is loaded in the HMA
28304                                                     ;     (DOS=HIGH)
28305
28306 00001421 83FFFF                     cmp     di,0FFFFh
28307 00001424 7506                       jne     short got_hma
28308
28309                                     ;mov    di,0            ; dont xor di,di Z flag needed
28310                                     ; 05/09/2023
28311                                     ; zf=1
28312 00001426 47                         inc     di ; 0FFFFh -> 0
28313                                     ; zf=1
28314
28315                                     ;zf=1
28316                                     ;xor    di,di   ; 25/10/2022
28317                                     ;zf=1
28318 00001427 2E8E06[6403]               mov     es,[cs:memhi]
28319                                   got_hma:
28320 0000142C 5A                         pop     dx
28321 0000142D 5B                         pop     bx
28322 0000142E C3                         retn
28323
28324                                     ; 11/04/2024 - Retro DOS v5.0
28325                                     ; PCDOS 7.1 IBMBIO.COM
28326                                     ;;;
28327                                   gba_2:
28328 0000142F C43F                       les     di,[bx]
28329 00001431 09FF                       or      di,di
28330                                     ;pop    dx
28331                                     ;pop    bx
28332                                     ;retn
28333                                     ; 11/04/2024 - Retro DOS v5.0
28334 00001433 EBF7                       jmp     short got_hma
28335                                     ;;;
28336
28337                                   ; ----------------------------------------------------------------------
28338
28339                                   set_buffer_info:
28340
28341                                   ;function: set buf_link,buf_id,buf_sector
28342                                   ;
28343                                   ;in: es:di -> buffer header to be set.
28344                                   ;    ax = di
28345                                   ;
28346                                   ;out:
28347                                   ;    above entries set.
28348
28349                                       ; 25/10/2022
28350 00001435 2EFF36[BD02]               push    word [cs:buf_prev_off]
28351                                     ;pop    word [es:di+buffinfo.buf_prev]
28352 0000143A 268F4502                   pop     word [es:di+2]
28353 0000143E 2EA3[BD02]                 mov     [cs:buf_prev_off],ax
28354 00001442 2E0306[9D02]               add     ax,[cs:singlebuffersize]     ;adjust ax
28355                                     ;mov    [es:di+buffinfo.buf_next],ax
28356 00001447 268905                     mov     [es:di],ax
28357                                     ;mov    word [es:di+buffinfo.buf_ID],00FFh  ; new buffer free
28358 0000144A 26C74504FF00               mov     word [es:di+4],00FFh
28359                                     ;mov    word [es:di+buffinfo.buf_sector],0   ; to compensate the masm 3 bug
28360 00001450 26C745060000               mov     word [es:di+6],0
28361                                     ;mov    word [es:di+buffinfo.buf_sector+2],0 ; to compensate the masm 3 bug
28362 00001456 26C745080000               mov     word [es:di+8],0
28363 0000145C C3                         retn
28364
28365                                   ; ==========================================================================
28366                                   ; MSSTACK initialization routine - MSDOS 6.0 - SYSDINIT1.ASM - 1991
28367                                   ; --------------------------------------------------------------------------
28368                                   ; 27/03/2019 - Retro DOS v4.0
28369
28370                                   ; ----------------------------------------------------------------------
28371                                   ; ibmstack initialization routine.
28372                                   ;
28373                                   ;   to follow the standard interrupt sharing scheme, msstack.asm
28374                                   ;   has been modified. this initialization routine also has to
```

```
28375                                   ;   be modified because for the interrupt level 7 and 15, firstflag
28376                                   ;   should be set to signal that this interrupt handler is the
28377                                   ;   first handler hooked to this interrupt vector.
28378                                   ;   we determine this by looking at the instruction pointed by
28379                                   ;   this vector. if it is iret, then this handler should be the
28380                                   ;   first one. in our case, only the interrupt vector 77h is the
28381                                   ;   interrupt level 15. (we don't hook interrupt level 7.)
28382                                   ;
28383                                   ;   the followings are mainly due to m.r.t; ptm fix of p886 12/3/86
28384                                   ;   some design changes are needed to the above interrupt sharing
28385                                   ;   method. the above sharing scheme assumes that 1). interrupt
28386                                   ;   sharing is never done on levels that have bios support. 2). "phantom"
28387                                   ;   interrupts would only be generated on levels 7 and 15.
28388                                   ;   these assumptions are not true any more. we have to use the firstflag
28389                                   ;   for every level of interrupt. we will set the firstflag on the following
28390                                   ;   conditions:
28391                                   ;
28392                                   ;    a.      if the cs portion of the vector is 0000, then "first"
28393                                   ;    b. else if cs:ip points to valid shared header, then not "first"
28394                                   ;    c. else if cs:ip points to an iret, then "first"
28395                                   ;    d. else if cs:ip points to dummy, then "first"
28396                                   ;
28397                                   ;   where dummy is - the cs portion must be f000, and the ip portion must
28398                                   ;   be equal to the value at f000:ff01. this location is the initial value
28399                                   ;   from vector_table for interrupt 7, one of the preserved addresses in all
28400                                   ;   the bioses for all of the machines.
28401                                   ;
28402                                   ;   system design group requests bios to handle the phantom interrupts.
28403                                   ;
28404                                   ;   the "phantom" interrupt is an illegal interrupt such as an interrupt
28405                                   ;   produced by the bogus adapter card even without interrupt request is
28406                                   ;   set.  more specifically, 1). the 8259 has a feature when running in
28407                                   ;   edge triggered mode to latch a pulse and present the interrupt when
28408                                   ;   the processor indicates interrupt acknowledge (inta). the interrupt
28409                                   ;   pulse was exist at the time of inta to get a "phantom" interrupt.
28410                                   ;   2). or, this is caused by adapter cards placing a glitch on the
28411                                   ;   interrupt line.
28412                                   ;
28413                                   ;   to handle those "phantom" interrupts, the main stack code will check
28414                                   ;   the own firstflag, and if it is not "first" (which means the forward
28415                                   ;   pointer points to the legal shared interrupt handler), then pass the
28416                                   ;   control. if it is the first, then the following action should be
28417                                   ;   taken. we don't have to implement skack logic in this case.
28418                                   ;
28419                                   ;   to implement this logic, we rather choose a simple method.
28420                                   ;   if ont of the above "firstflag" conditions is met, we are not
28421                                   ;   going to hook this interrupt vector. the reason is if the original
28422                                   ;   vector points to "iret" and do nothing, we don't need
28423                                   ;   to implement the stack logic for it. this will simplify implementation
28424                                   ;   while maintaining compatibility with the old version of dos.
28425                                   ;   this implies that in the main stack code, there might be a stack code
28426                                   ;   that will never be used, a dead code.
28427                                   ;
28428                                   ;in - cs, ds -> sysinitseg, es -> relocated stack code & data.
28429
28430                                       ; 25/10/2022 - Retro DOS 4.0 (Modified MSDOS 5.0 IO.SYS, SYSINIT)
28431                                       ; (SYSINIT:1287h)
28432
28433                                       ; 11/04/2024 - Retro DOS 5.0 (Modified PCDOS 7.1 IBMBIO.COM)
28434                                       ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:157Ch)
28435
28436                                       ; 14/12/2022
28437                                   stackinit:
28438 0000145D 50                         push  ax
28439 0000145E 1E                         push  ds
28440 0000145F 06                         push  es
28441 00001460 53                         push  bx
28442 00001461 51                         push  cx
28443 00001462 52                         push  dx
28444 00001463 57                         push  di
28445 00001464 56                         push  si
28446 00001465 55                         push  bp
28447
28448                                   ;currently es -> stack code area
28449
28450                                       ; 12/12/2022
28451                                       ; ds = cs
28452 00001466 A1[8C02]                   mov   ax,[stack_count]
28453 00001469 89C1                       mov   cx,ax  ; *!*!*
28454                                     ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
28455                                     ; (MSDOS 5.0 IO.SYS - SYSINIT:1290h)
28456                                     ;mov   ax,[cs:stack_count] ; !!     ;defined in cs
28457 0000146B 26A3[0200]                 mov   [es:stackcount],ax          ;defined in stack code area
28458                                     ; (MSDOS 5.0 IO.SYS - SYSINIT:1298h)
28459 0000146F A1[8E02]                   mov   ax,[stack_size]       ; !!          ;in cs
28460 00001472 26A3[0600]                 mov   [es:stacksize],ax
28461                                     ; 12/12/2022
28462 00001476 A1[9002]                   mov   ax,[stack_addr]                     ; offset
28463                                     ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
28464                                     ; (MSDOS 5.0 IO.SYS - SYSINIT:129Fh)
28465                                     ;mov   ax,[cs:stack_addr]  ; !!
28466 00001479 26A3[0800]                 mov   [es:stacks],ax
28467                                     ; 12/12/2022
28468 0000147D 89C5                       mov   bp,ax ; *!*
28469 0000147F A1[9202]                   mov   ax,[stack_addr+2]
28470                                     ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
28471                                     ; (MSDOS 5.0 IO.SYS - SYSINIT:129Fh)
28472                                     ;mov   ax,[cs:stack_addr+2] ; !!    ; segment
28473 00001482 26A3[0A00]                 mov   [es:stacks+2],ax
28474
28475                                   ; initialize the data fields with the parameters
28476
28477                                   ; "firstentry" will always be at stacks
28478
28479                                       ;mov   bp,[es:stacks]               ; get offset of stack
28480                                       ; 12/12/2022
28481                                       ; bp = [es:stacks] ; *!*
28482 00001486 26892E[0C00]               mov   [es:firstentry],bp
28483
28484                                   ; the stacks will always immediately follow the table entries
28485
28486 0000148B B80800                      mov   ax,entrysize ; 8
28487                                     ;mov   cx,[es:stackcount]
28488                                     ; 12/12/2022
28489                                     ; cx = [es:stackcount] ; *!*!*
28490 0000148E F7E1                       mul   cx
28491 00001490 01E8                       add   ax,bp
28492 00001492 26A3[0400]                 mov   [es:stackat],ax
28493 00001496 89C3                       mov   bx,ax
28494 00001498 83EB02                     sub   bx,2
28495
28496                                   ; zero the entire stack area to start with
28497
28498 0000149B 268B3E[0400]               mov   di,[es:stackat]
```

```
28499 000014A0 26A1[0600]                mov     ax,[es:stacksize]
28500 000014A4 F7E1                      mul     cx
28501 000014A6 89C1                      mov     cx,ax
28502 000014A8 31C0                      xor     ax,ax
28503 000014AA 06                        push    es
28504 000014AB 1F                        pop     ds                              ;ds = relocated stack code seg.
28505
28506                                ;now, ds -> stack code area
28507
28508 000014AC 8E06[0A00]                mov     es,[stacks+2]                   ; get segment of stack area.
28509 000014B0 FC                        cld
28510 000014B1 F3AA                      rep     stosb
28511
28512 000014B3 8B0E[0200]                mov     cx,[stackcount]
28513
28514                                ; loop for "count" times, building a table entry
28515                                ;  cs = sysinitseg, ds = relocated stack code seg, es = segment of stack space
28516                                ;  cx = number of entries
28517                                ;  es:bp => base of stacks - 2
28518                                ;  es:bx => first table entry
28519
28520                                buildloop:
28521                                    ; 11/12/2022
28522                                    ;mov   byte [es:bp+allocbyte],free   ; mov [es:bp+0],0
28523                                    ; 25/10/2022
28524                                    ;mov   byte [es:bp],free
28525                                    ; 06/07/2023
28526 000014B7 26884600                 mov     [es:bp],al ; 0 ; free
28527 000014BB 26884601                 mov     [es:bp+intlevel],al   ; ax = 0
28528                                   ;mov    [es:bp+1],al
28529 000014BF 26894602                 mov     [es:bp+savedsp],ax
28530                                   ;mov    [es:bp2],ax
28531 000014C3 26894604                 mov     [es:bp+savedss],ax
28532                                   ;mov    [es:bp+4],ax
28533 000014C7 031E[0600]               add     bx,[stacksize]
28534 000014CB 26895E06                 mov     [es:bp+newsp],bx             ; mov [es:bp+6],bx
28535                                   ;mov    [es:bp+6],bx
28536 000014CF 26892F                   mov     [es:bx],bp
28537 000014D2 83C508                   add     bp,entrysize ; 8
28538
28539 000014D5 E2E0                     loop    buildloop
28540
28541 000014D7 83ED08                   sub     bp,entrysize ; 8
28542 000014DA 892E[0E00]               mov     [lastentry],bp
28543 000014DE 892E[1000]               mov     [nextentry],bp
28544
28545 000014E2 1E                       push    ds
28546                                   ;mov    ax,0F000h               ;look at the model byte
28547                                   ; 05/09/2023
28548 000014E3 B4F0                     mov     ah,0F0h ; ax = 0F000h
28549 000014E5 8ED8                     mov     ds,ax
28550 000014E7 803EFEFFF9               cmp     byte [0FFFEh],0F9h ; mdl_convert ; convertible?
28551 000014EC 1F                       pop     ds
28552 000014ED 7504                     jne     short skip_disablenmis
28553
28554 000014EF B007                      mov     al,07h                  ; disable convertible nmis
28555 000014F1 E672                      out     72h,al
28556
28557                                skip_disablenmis:
28558 000014F3 31C0                      xor     ax,ax
28559 000014F5 8EC0                      mov     es,ax                   ;es - segid of vector table at 0
28560                                                                    ;ds - relocated stack code segment
28561 000014F7 FA                        cli
28562
28563                                    ;irp    aa,<02,08,09,70>
28564                                    ;
28565                                    ;mov    si,aa&h*4               ;pass where vector is to be adjusted
28566                                    ;mov    di,offset int19old&aa ;we have to set old&aa for int19 handler too.
28567                                    ;mov    bx,offset old&aa        ;pass where to save original owner pointer
28568                                    ;mov    dx,offset int&aa        ;pass where new handler is
28569                                    ;call   new_init_loop           ;adjust the vector to new handler,
28570                                    ;                               ;saving pointer to original owner
28571                                    ;endm
28572
28573                                stkinit_02:
28574 000014F8 BE0800                    mov     si,02h*4 ; 8
28575 000014FB BF[B305]                  mov     di,INT19OLD02
28576 000014FE BB[1200]                  mov     bx,old02
28577 00001501 BA[1600]                  mov     dx,int02
28578 00001504 E84801                    call    new_init_loop
28579                                stkinit_08:
28580 00001507 BE2000                    mov     si,08h*4 ; 32
28581 0000150A BF[B805]                  mov     di,INT19OLD08
28582 0000150D BB[3800]                  mov     bx,old08
28583 00001510 BA[3C00]                  mov     dx,int08
28584 00001513 E83901                    call    new_init_loop
28585                                stkinit_09:
28586 00001516 BE2400                    mov     si,09h*4 ; 36
28587 00001519 BF[BD05]                  mov     di,INT19OLD09
28588 0000151C BB[4100]                  mov     bx,old09
28589 0000151F BA[4500]                  mov     dx,int09
28590 00001522 E82A01                    call    new_init_loop
28591                                stkinit_70:
28592 00001525 BEC001                    mov     si,70h*4 ; 448
28593 00001528 BF[DB05]                  mov     di,INT19OLD70
28594 0000152B BB[4E00]                  mov     bx,old70
28595 0000152E BA[5200]                  mov     dx,int70
28596 00001531 E81B01                    call    new_init_loop
28597
28598                                    ;irp    aa,<0a,0b,0c,0d,0e,72,73,74,76,77> ;shared interrupts
28599                                    ;
28600                                    ;mov    si,aa&h*4               ;pass where vector is to be adjusted
28601                                    ;push   ds                      ;save relocated stack code segment
28602                                    ;lds    bx, es:[si]             ;ds:bx -> original interrupt handler
28603                                    ;push   ds
28604                                    ;pop    dx                      ;dx = segment value
28605                                    ;
28606                                    ;cmp    dx,0
28607                                    ;jz     int&aa&_first
28608                                    ;
28609                                    ;cmp    byte ptr ds:[bx],0cfh ;does vector point to an iret?
28610                                    ;jz     int&aa&_first
28611                                    ;
28612                                    ;cmp    word ptr ds:[bx.6],424bh ;magic offset (see int&aa, msstack.inc)
28613                                    ;jz     int&aa&_not_first
28614                                    ;
28615                                    ;cmp    dx,0f000h               ;rom bios segment
28616                                    ;jnz    int&aa&_not_first
28617                                    ;
28618                                    ;push   es
28619                                    ;push   dx
28620                                    ;mov    dx,0f000h
28621                                    ;mov    es,dx
28622                                    ;cmp    bx,word ptr es:0ff01h
```

```
                                        ;pop    dx
                                    ;pop    es
                                    ;jz     int&aa&_first
                                    ;
                                ;int&aa&_not_first:                     ;not the first. we are going to hook vector.
                                    ;pop    ds
                                    ;mov    di, offset int19old&aa;we have to set old&aa for int19 handler too.
                                    ;mov    bx, offset old&aa     ;pass where to save original owner pointer
                                    ;mov    dx, offset int&aa     ;pass where new handler is
                                    ;call   new_init_loop         ;adjust the vector to new handler, saving
                                    ;                             ;pointer to original owner.
                                    ;jmp    short int&aa&_end
                                ;int&aa&_first:                         ;the first. don't have to hook stack code.
                                    ;pop    ds
                                ;int&aa&_end:
                                    ;
                                    ;endm

                                stkinit_0A:
00001534 BE2800                     mov     si,0Ah*4 ; 40

                                ; 14/12/2022
                                %if 0
                                    ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
                                    push    ds

                                    lds     bx,[es:si]
                                    push    ds
                                    pop     dx

                                    cmp     dx,0
                                    je      short int_0A_first

                                    cmp     byte [bx],0CFh
                                    je      short int_0A_first

                                    cmp     word [bx+6],424Bh
                                    je      short int_0A_not_first

                                    cmp     dx,0F000h
                                    jne     short int_0A_not_first

                                    push    es
                                    push    dx
                                    mov     dx,0F000h
                                    mov     es,dx
                                    cmp     bx,[es:0FF01h]
                                            pop     dx
                                    pop     es
                                    je      short int_0A_first
                                %Endif

                                    ; 14/12/2022
                                    ; 25/10/2022
00001537 E8EB00                     call    int_xx_first_check ; 27/03/2019 - Retro DOS v4.0
0000153A 730C                       jnc     short int_0A_first

                                int_0A_not_first:
                                    ; 14/12/2022
                                    ; 25/10/2022
                                    ;pop    ds
0000153C BF[C205]                   mov     di,INT19OLD0A
0000153F BB[5900]                   mov     bx,old0A
00001542 BA[5700]                   mov     dx,int0A
00001545 E80701                     call    new_init_loop

                                    ; 14/12/2022
                                    ;jmp    short int_0A_end
                                ;int_0A_first:
                                    ; 25/10/2022
                                    ;pop    ds

                                    ; 14/12/2022
                                int_0A_first:
                                int_0A_end:

                                stkinit_0B:
00001548 BE2C00                     mov     si,0Bh*4 ; 44

                                    ; 14/12/2022
                                    ; 25/10/2022
0000154B E8D700                     call    int_xx_first_check ; 27/03/2019 - Retro DOS v4.0
0000154E 730C                       jnc     short int_0B_end ; int_0B_first

                                ; 14/12/2022
                                %if 0
                                    ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
                                    push    ds
                                    lds     bx,[es:si]
                                    push    ds
                                    pop     dx

                                    cmp     dx,0
                                    je      short int_0B_first

                                    cmp     byte [bx],0CFh
                                    je      short int_0B_first

                                    cmp     word [bx+6],424Bh
                                    je      short int_0B_not_first

                                    cmp     dx,0F000h
                                    jne     short int_0B_not_first

                                    push    es
                                    push    dx
                                    mov     dx,0F000h
                                    mov     es,dx
                                    cmp     bx,[es:0FF01h]
                                    pop     dx
                                    pop     es
                                    je      short int_0B_first
                                %endif

                                int_0B_not_first:
                                    ; 14/12/2022
                                    ; 25/10/2022
                                    ;pop    ds
00001550 BF[C705]                   mov     di,INT19OLD0B
00001553 BB[7100]                   mov     bx,old0B
00001556 BA[6F00]                   mov     dx,int0B
00001559 E8F300                     call    new_init_loop

                                    ; 14/12/2022
```

```
                                    ;jmp    short int_0B_end
                                ;int_0B_first:
                                    ; 25/10/2022
                                    ;pop    ds

                                int_0B_end:

                                stkinit_0C:
0000155C BE3000                     mov     si,0Ch*4 ; 48

                                    ; 14/12/2022
                                    ; 25/10/2022
0000155F E8C300                     call    int_xx_first_check
00001562 730C                       jnc     short int_0C_end ; int_0C_first

                                ; 14/12/2022
                                %if 0
                                    ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
                                    push    ds
                                    lds     bx,[es:si]
                                    push    ds
                                    pop     dx

                                    cmp     dx,0
                                    je      short int_0C_first

                                    cmp     byte [bx],0CFh
                                    je      short int_0C_first

                                    cmp     word [bx+6],424Bh
                                    je      short int_0C_not_first

                                    cmp     dx,0F000h
                                    jne     short int_0C_not_first

                                    push    es
                                    push    dx
                                    mov     dx,0F000h
                                    mov     es,dx
                                    cmp     bx,[es:0FF01h]
                                    pop     dx
                                    pop     es
                                    je      short int_0C_first
                                %endif

                                int_0C_not_first:
                                    ; 14/12/2022
                                    ; 25/10/2022
                                    ;pop    ds
00001564 BF[CC05]                   mov     di,INT19OLD0C
00001567 BB[8900]                   mov     bx,old0C
0000156A BA[8700]                   mov     dx,int0C
0000156D E8DF00                     call    new_init_loop

                                    ; 14/12/2022
                                    ;jmp    short int_0C_end
                                ;int_0C_first:
                                    ; 25/10/2022
                                    ;pop    ds

                                int_0C_end:

                                stkinit_0D:
00001570 BE3400                     mov     si,0Dh*4 ; 52

                                    ; 14/12/2022
                                    ; 25/10/2022
00001573 E8AF00                     call    int_xx_first_check
00001576 730C                       jnc     short int_0D_end ; int_0D_first

                                ; 14/12/2022
                                %if 0
                                    ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
                                    push    ds
                                    lds     bx,[es:si]
                                    push    ds
                                    pop     dx

                                    cmp     dx,0
                                    je      short int_0D_first

                                    cmp     byte [bx],0CFh
                                    je      short int_0D_first

                                    cmp     word [bx+6],424Bh
                                    je      short int_0D_not_first

                                    cmp     dx,0F000h
                                    jne     short int_0D_not_first

                                    push    es
                                    push    dx
                                    mov     dx,0F000h
                                    mov     es,dx
                                    cmp     bx,[es:0FF01h]
                                    pop     dx
                                    pop     es
                                    je      short int_0D_first
                                %endif

                                int_0D_not_first:
                                    ; 14/12/2022
                                    ; 25/10/2022
                                    ;pop    ds
00001578 BF[D105]                   mov     di,INT19OLD0D
0000157B BB[A100]                   mov     bx,old0D
0000157E BA[9F00]                   mov     dx,int0D
00001581 E8CB00                     call    new_init_loop

                                    ; 14/12/2022
                                    ;jmp    short int_0D_end
                                    ; 02/11/2022
                                ;int_0D_first:
                                    ;pop    ds

                                int_0D_end:

                                stkinit_0E:
00001584 BE3800                     mov     si,0Eh*4 ; 56

                                    ; 14/12/2022
                                    ; 25/10/2022
00001587 E89B00                     call    int_xx_first_check
0000158A 730C                       jnc     short int_0E_end ; int_0E_first
```

```
                                    ; 14/12/2022
                                    %if 0
                                        ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
                                        push    ds
                                        lds     bx,[es:si]
                                        push    ds
                                        pop     dx

                                        cmp     dx,0
                                        je      short int_0E_first

                                        cmp     byte [bx],0CFh
                                        je      short int_0E_first

                                        cmp     word [bx+6],424Bh
                                        je      short int_0E_not_first

                                        cmp     dx,0F000h
                                        jne     short int_0E_not_first

                                        push    es
                                        push    dx
                                        mov     dx,0F000h
                                        mov     es,dx
                                        cmp     bx,[es:0FF01h]
                                        pop     dx
                                        pop     es
                                        je      short int_0E_first
                                    %endif

                                    int_0E_not_first:
                                        ; 14/12/2022
                                        ; 25/10/2022
                                        ;pop    ds
0000158C BF[D605]                       mov     di,INT19OLD0E
0000158F BB[B900]                       mov     bx,old0E
00001592 BA[B700]                       mov     dx,int0E
00001595 E8B700                         call    new_init_loop

                                        ; 14/12/2022
                                        ;jmp    short int_0E_end
                                    ;int_0E_first:
                                        ; 25/10/2022
                                        ;pop    ds

                                    int_0E_end:

                                    stkinit_72:
00001598 BEC801                         mov     si,72h*4 ; 456

                                        ; 14/12/2022
                                        ; 25/10/2022
0000159B E88700                         call    int_xx_first_check
0000159E 730C                           jnc     short int_72_end ; int_72_first

                                    ; 14/12/2022
                                    %if 0
                                        ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
                                        push    ds
                                        lds     bx,[es:si]
                                        push    ds
                                        pop     dx

                                        cmp     dx,0
                                        je      short int_72_first

                                        cmp     byte [bx],0CFh
                                        je      short int_72_first

                                        cmp     word [bx+6],424Bh
                                        je      short int_72_not_first

                                        cmp     dx,0F000h
                                        jne     short int_72_not_first

                                        push    es
                                        push    dx
                                        mov     dx,0F000h
                                        mov     es,dx
                                        cmp     bx,[es:0FF01h]
                                        pop     dx
                                        pop     es
                                        je      short int_72_first
                                    %endif

                                    int_72_not_first:
                                        ; 14/12/2022
                                        ; 25/10/2022
                                        ;pop    ds
000015A0 BF[E005]                       mov     di,INT19OLD72
000015A3 BB[D100]                       mov     bx,old72
000015A6 BA[CF00]                       mov     dx,int72
000015A9 E8A300                         call    new_init_loop

                                        ; 14/12/2022
                                        ;jmp    short int_72_end
                                    ;int_72_first:
                                        ; 25/10/2022
                                        ;pop    ds

                                    int_72_end:

                                    stkinit_73:
000015AC BECC01                         mov     si,73h*4 ; 460

                                        ; 14/12/2022
                                        ; 25/10/2022
000015AF E87300                         call    int_xx_first_check
000015B2 730C                           jnc     short int_73_end ; int_73_first

                                    ; 14/12/2022
                                    %if 0
                                        ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
                                        push    ds
                                        lds     bx,[es:si]
                                        push    ds
                                        pop     dx

                                        cmp     dx,0
                                        je      short int_73_first

                                        cmp     byte [bx],0CFh
                                        je      short int_73_first
```

```
                                    cmp     word [bx+6],424Bh
                                    je      short int_73_not_first

                                    cmp     dx,0F000h
                                    jne     short int_73_not_first

                                    push    es
                                    push    dx
                                    mov     dx,0F000h
                                    mov     es,dx
                                    cmp     bx,[es:0FF01h]
                                    pop     dx
                                    pop     es
                                    je      short int_73_first
                                %endif

                                int_73_not_first:
                                    ; 14/12/2022
                                    ; 25/10/2022
                                    ;pop    ds
000015B4 BF[E505]                   mov     di,INT19OLD73
000015B7 BB[E900]                   mov     bx,old73
000015BA BA[E700]                   mov     dx,int73
000015BD E88F00                     call    new_init_loop

                                    ; 14/12/2022
                                    ;jmp    short int_73_end
                                ;int_73_first:
                                    ; 25/10/2022
                                    ;pop    ds

                                int_73_end:

                                stkinit_74:
000015C0 BED001                     mov     si,74h*4 ; 464

                                    ; 14/12/2022
                                    ; 25/10/2022
000015C3 E85F00                     call    int_xx_first_check
000015C6 730C                       jnc     short int_74_end ; int_74_first

                                ; 14/12/2022
                                %if 0
                                    ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
                                    push    ds
                                    lds     bx,[es:si]
                                    push    ds
                                    pop     dx

                                    cmp     dx,0
                                    je      short int_74_first

                                    cmp     byte [bx],0CFh
                                    je      short int_74_first

                                    cmp     word [bx+6],424Bh
                                    je      short int_74_not_first

                                    cmp     dx,0F000h
                                    jne     short int_74_not_first

                                    push    es
                                    push    dx
                                    mov     dx,0F000h
                                    mov     es,dx
                                    cmp     bx,[es:0FF01h]
                                    pop     dx
                                    pop     es
                                    je      short int_74_first
                                %endif

                                int_74_not_first:
                                    ; 14/12/2022
                                    ; 25/10/2022
                                    ;pop    ds
000015C8 BF[EA05]                   mov     di,INT19OLD74
000015CB BB[0101]                   mov     bx,old74
000015CE BA[FF00]                   mov     dx,int74
000015D1 E87B00                     call    new_init_loop

                                    ; 14/12/2022
                                    ;jmp    short int_74_end
                                ;int_74_first:
                                    ; 25/10/2022
                                    ;pop    ds

                                int_74_end:

                                stkinit_76:
000015D4 BED801                     mov     si,76h*4 ; 472

                                    ; 14/12/2022
                                    ; 25/10/2022
000015D7 E84B00                     call    int_xx_first_check
000015DA 730E                       jnc     short int_76_end ; int_76_first

                                ; 14/12/2022
                                %if 0
                                    ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
                                    push    ds
                                    lds     bx,[es:si]
                                    push    ds
                                    pop     dx

                                    cmp     dx,0
                                    je      short int_76_first

                                    cmp     byte [bx],0CFh
                                    je      short int_76_first

                                    cmp     word [bx+6],424Bh
                                    je      short int_76_not_first

                                    cmp     dx,0F000h
                                    jne     short int_76_not_first

                                    push    es
                                    push    dx
                                    mov     dx,0F000h
                                    mov     es,dx
                                    cmp     bx,[es:0FF01h]
                                    pop     dx
                                    pop     es
```

```
                                    je      short int_76_first
                                %endif

                                int_76_not_first:
                                    ; 14/12/2022
                                    ; 25/10/2022
                                    ;pop    ds
000015DC BF[EF05]                   mov     di,INT19OLD76
000015DF BB[1901]                   mov     bx,old76
000015E2 BA[1701]                   mov     dx,int76
000015E5 E86700                     call    new_init_loop

                                    ; 14/12/2022
000015E8 EB00                       jmp     short int_76_end
                                ;int_76_first:
                                    ; 25/10/2022
                                    ;pop    ds

                                int_76_end:

                                stkinit_77:
000015EA BEDC01                     mov     si,77h*4 ; 476

                                    ; 14/12/2022
                                    ; 25/10/2022
000015ED E83500                     call    int_xx_first_check
000015F0 730C                       jnc     short int_77_end ; int_77_first

                                ; 14/12/2022
                                %if 0
                                    ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
                                    push    ds
                                    lds     bx,[es:si]
                                    push    ds
                                    pop     dx

                                    cmp     dx,0
                                    je      short int_77_first

                                    cmp     byte [bx],0CFh
                                    je      short int_77_first

                                    cmp     word [bx+6],424Bh
                                    je      short int_77_not_first

                                    cmp     dx,0F000h
                                    jne     short int_77_not_first

                                    push    es
                                    push    dx
                                    mov     dx,0F000h
                                    mov     es,dx
                                    cmp     bx,[es:0FF01h]
                                    pop     dx
                                    pop     es
                                    je      short int_77_first
                                %endif

                                int_77_not_first:
                                    ; 14/12/2022
                                    ; 25/10/2022
                                    ;pop    ds
000015F2 BF[F405]                   mov     di,INT19OLD77
000015F5 BB[3101]                   mov     bx,old77
000015F8 BA[2F01]                   mov     dx,int77
000015FB E85100                     call    new_init_loop

                                    ; 14/12/2022
                                    ;jmp    short int_77_end
                                ;int_77_first:
                                    ; 25/10/2022
                                    ;pop    ds

                                int_77_end:
000015FE 1E                         push    ds
000015FF B800F0                     mov     ax,0F000h               ; look at the model byte
00001602 8ED8                       mov     ds,ax
00001604 803EFEFFF9                 cmp     byte [0FFFEh],0F9h ; mdl_convert ; pc convertible?
00001609 1F                         pop     ds
0000160A 7504                       jne     short skip_enablenmis

0000160C B027                       mov     al,27h                  ; enable convertible nmis
0000160E E672                       out     72h,al

                                ; 25/10/2022
                                ; (MSDOS 5.0 SYSINIT:15FBh)

                                skip_enablenmis:
00001610 FB                         sti
                                    ;;mov   ax,Bios_Data ; 70h
                                    ;mov    ax,KERNEL_SEGMENT ; 70h
                                    ; 21/10/2022
00001611 B87000                     mov     ax,DOSBIODATASEG ; 0070h
00001614 8ED8                       mov     ds,ax

                                    ;mov    [640h],1 ; SYSINIT:1736h for MSDOS 6.21 IO.SYS

00001616 C606[B105]01               mov     byte [INT19SEM],1       ; indicate that int 19
                                                                    ; initialization is complete

0000161B 5D                         pop     bp                      ; restore all
0000161C 5E                         pop     si
0000161D 5F                         pop     di
0000161E 5A                         pop     dx
0000161F 59                         pop     cx
00001620 5B                         pop     bx
00001621 07                         pop     es
00001622 1F                         pop     ds
00001623 58                         pop     ax
00001624 C3                         retn

                                ; 14/12/2022
                                ; ----------------------------------------------------------------------

                                    ; 14/12/2022
                                    ; 25/10/2022
                                ;%if 0
                                    ; 27/03/2019 - Retro DOS v4.0
                                int_xx_first_check:
00001625 1E                         push    ds
00001626 26C51C                     lds     bx,[es:si]
00001629 1E                         push    ds
0000162A 5A                         pop     dx
```

```
                                    ;cmp    dx,0
                                    ;je     short int_xx_first
                                    ; 05/09/2023
0000162B 21D2                       and     dx,dx
0000162D 741E                       jz      short int_xx_first

0000162F 803FCF                     cmp     byte [bx],0CFh
00001632 7419                       je      short int_xx_first

00001634 817F064B42                 cmp     word [bx+6],424Bh
00001639 7411                       je      short int_xx_not_first

0000163B 81FA00F0                   cmp     dx,0F000h
0000163F 750B                       jne     short int_xx_not_first

00001641 06                         push    es
                                    ;push   dx
                                    ;mov    dx,0F000h
00001642 8EC2                       mov     es,dx
00001644 263B1E01FF                 cmp     bx,[es:0FF01h]
                                            ;pop    dx
00001649 07                         pop     es
0000164A 7401                       je      short int_xx_first

                                int_xx_not_first:
0000164C F9                         stc
                                int_xx_first:
0000164D 1F                         pop     ds
0000164E C3                         retn

                                ;%endif

                                ; ----------------------------------------------------------------------
                                ; 27/03/2019 - Retro DOS v4.0

                                ; 25/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS, SYSINIT)
                                ; (SYSINIT:1610h)

                                ; 13/04/2024 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM, SYSINIT)
                                ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:1905h)

                                new_init_loop:
                                    ;;; 13/04/2024 (PCDOS 7.1 IBMBIO.COM)
0000164F 2E803E[6E03]02             cmp     byte [cs:dosdata_umb],2
                                            ; is DOSDATA=UMB done ? (DOSDATA is in UMB)
00001655 7510                       jne     short new_init_loop_1st
00001657 1E                         push    ds              ; restore original/previous interrupt handler
                                            ; (from int19old?? field in BIOSDATA)
                                    ;mov    ax,70h
00001658 B87000                     mov     ax,DOSBIODATASEG
0000165B 8ED8                       mov     ds,ax
0000165D C505                       lds     ax,[di]         ; restore original Int ?? handler addr from int19old?? field
0000165F 268904                     mov     [es:si],ax      ; copy the original int handler addr to its int vector addr
00001662 268C5C02                   mov     [es:si+2],ds
00001666 1F                         pop     ds
                                new_init_loop_1st:
                                    ;;;

                                ;input: si=ofset into vector table of the particular int vector being adjusted
                                ;   bx=ds:offset of oldxx, where will be saved the pointer to original owner
                                ;   dx=ds:offset of intxx, the new interrupt handler
                                ;   di=offset value of int19old&aa variable in bios.
                                ;   es=zero, segid of vector table
                                ;   ds=relocated stack code segment

                                ; 13/04/2024
                                %if 0
                                    mov     ax,[es:si]              ;remember offset in vector
                                    mov     [bx],ax                 ; to original owner in ds
                                    mov     ax,[es:si+2]            ;remember segid in vector
                                    mov     [bx+2],ax               ; to original owner in ds

                                    push    ds
                                    ;;mov   ax,Bios_Data ; 70h
                                    ;mov    ax,KERNEL_SEGMENT ; 70h
                                    ; 21/10/2022
                                    mov     ax,DOSBIODATASEG ; 0070h
                                    mov     ds,ax                   ;set int19oldxx value in bios for
                                    mov     ax,[es:si]              ;int 19 handler
                                    mov     [di],ax
                                    mov     ax,[es:si+2]
                                    mov     [di+2],ax
                                    pop     ds
                                %else
                                    ; 13/04/2024 - Retro DOS v5.0
0000167F 1E                         push    ds
00001668 268B4402                   mov     ax,[es:si+2]            ;remember segid in vector
0000166C 894702                     mov     [bx+2],ax               ; to original owner in ds
0000166F 50                         push    ax
00001670 268B04                     mov     ax,[es:si]              ;remember offset in vector
00001673 8907                       mov     [bx],ax                 ; to original owner in ds
00001675 50                         push    ax
00001676 B87000                     mov     ax,DOSBIODATASEG ; 0070h
00001679 8ED8                       mov     ds,ax                   ;set int19oldxx value in bios for
0000167B 58                         pop     ax                      ;int 19 handler
0000167C 8905                       mov     [di],ax
0000167E 58                         pop     ax
0000167F 894502                     mov     [di+2],ax
00001682 1F                         pop     ds
                                %endif
00001683 268914                     mov     [es:si],dx      ;set vector to point to new int handler
00001686 268C5C02                   mov     [es:si+2],ds
0000168A C3                         retn

                                ; End of STACK initialization routine
                                ; ----------------------------------------------------------------------

                                ; ----------------------------------------------------------------------
                                ;set the devmark for mem command.
                                ;in: [memhi] - the address to place devmark
                                ;    [memlo] = 0
                                ;    al = id for devmark_id
                                ;out: devmark established.
                                ;     the address saved in cs:[devmark_addr]
                                ;     [memhi] increase by 1.
                                ; ----------------------------------------------------------------------

                                ; 25/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS, SYSINIT)
                                ; (SYSINIT:1637h)
                                ; 04/09/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS, SYSINIT)
                                ; (SYSINIT:176Ch)

                                ; 04/09/2023 - PCDOS 7.1 - IBMBIO.COM (SYSINIT:1944h)
```

```
                                setdevmark:

                                    ; 04/09/2023
                                    ;push   es
                                    ;push   cx

0000168B 2E8B0E[6403]               mov     cx,[cs:memhi]
00001690 2E890E[6719]               mov     [cs:devmark_addr],cx
00001695 8EC1                       mov     es,cx
                                    ; 25/10/2022
                                    ;mov    [es:devmark.id],al
00001697 26A20000                   mov     [es:0],al
0000169B 41                         inc     cx
                                    ;mov    [es:devmark.seg],cx
0000169C 26890E0100                 mov     [es:1],cx

                                    ; 04/09/2023
                                    ;pop    cx
                                    ;pop    es

000016A1 2EFF06[6403]               inc     word [cs:memhi]
000016A6 C3                         retn

                                ; ---------------------------------------------------------------------------
                                ; SYSPRE.ASM - MSDOS 6.0 - 1992
                                ; ---------------------------------------------------------------------------
                                ;; pre-load and final placement of dblspace.bin
                                ;
                                ; 08/04/2024 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
                                ; ============================================================

                                ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:1964h)
                                ;;; ---------------------------------------------------------------------------
000016A7 [AB16]                 MagicDDNamePtr:     dw MagicDDName ; "\DBLSPACE.BIN"
000016A9 433A                               db 'C:'
000016AB 5C44424C5350414345-    MagicDDName:        db '\DBLSPACE.BIN',0
000016B4 2E42494E00
000016B9 433A5C535441434B45-    StackerName:        db 'C:\STACKER.BIN',0
000016C2 522E42494E00
                                tiny_stub_start:
000016C8 FFFF                               dw 0FFFFh       ; phony device driver link
000016CA FFFF                               dw 0FFFFh       ; dw -1, -1
000016CC 0080                               dw 8000h        ; mark as character device for MEM display
000016CE 00000000                           dw 2 dup(0)     ; strategy and interrupt
000016D2 44424C5342494E24                   db 'DBLSBIN$'   ; magic default load
                                tiny_stub_end:                      ; (tiny_stub_end-tiny_stub_start = 18)

                                ; =============== S U B R O U T I N E =======================================

                                ; 08/04/2024 - Retro DOS v5.0
                                ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:1997h)

                                ;***MagicPreload - pre-load dblspace.bin
                                ;
                                ;   EXIT    ax = error code, 00 means none.
                                ;           ZF = true if ax == 0

                                MagicPreload:
                                    ; 13/04/2024 - Retro DOS v5.0
                                    ; ds = cs
                                    ;mov    byte [cs:setdevmarkflag],0 ; not for devmark
000016DA C606[6919]00               mov     byte [setdevmarkflag],0
000016DF E81031                     call    round
000016E2 0E                         push    cs
000016E3 07                         pop     es
                                    ;mov    byte [cs:DeviceHi],0  ; not to be loaded in UMB
                                    ; 13/04/2024
000016E4 C606[5124]00               mov     byte [DeviceHi],0
000016E9 E8791E                     call    InitDevLoad             ; set up sub-arena, DevLoadAddr,
                                                                    ;  DevLoadEnd, and DevEntry
                                                                    ;  gets arena name from bpb_addr
                                    ; 13/04/2024
                                    ; ds = cs

                                ;   check to make sure device driver fits our available space.

                                    ;mov    ax,[cs:DevLoadAddr]
000016EC A1[3524]                   mov     ax,[DevLoadAddr]
                                    ;add    ax,[cs:DevSize]             ; calculate seg after DD load
000016EF 0306[3324]                 add     ax,[DevSize]
000016F3 725E                       jc      short pre_exit_err      ; choke if overflows address space
                                    ;cmp    ax,[cs:DevLoadEnd]      ; does it overflow available space?
000016F5 3B06[3724]                 cmp     ax,[DevLoadEnd]
000016F9 7758                       ja      short pre_exit_err

                                _LoadDev:                           ; we're golden if not
                                    ; 13/04/2024
                                    ; ds = cs
                                    ;push   cs
                                    ;pop    ds
                                    ;mov    dx,[cs:MagicDDNamePtr]
000016FB 8B16[A716]                 mov     dx,[MagicDDNamePtr]
000016FF E8A41F                     call    ExecDev             ; load device driver using exec call
00001702 724F                       jb      short pre_exit_err

                                    ; 13/04/2024
                                    ; ds = cs
                                    ;les    bx,[cs:DevEntry]        ; point to the Magic DD header
00001704 C41E[3924]                 les     bx,[DevEntry]
00001708 26817F122C2E               cmp     word [es:bx+12h],2E2Ch; is it our stamp? ; ',.'
0000170E 7543                       jnz     short pre_exit_err
                                    ;mov    word [cs:MagicBackdoor],14h ; save the backdoor entry.
                                    ;                           ; (initial IP -EXE header offset 20-)
                                    ;mov    [cs:MagicBackdoor+2],es
00001710 C706[9003]1400             mov     word [MagicBackdoor],14h
00001716 8C06[9203]                 mov     [MagicBackdoor+2],es

0000171A 0E                         push    cs
0000171B 07                         pop     es
0000171C BB[6F03]                   mov     bx,packet

                                    ;mov    word [cs:break_addr],0
                                    ;mov    ax,[cs:DevLoadEnd]
                                    ;mov    [cs:break_addr+2],ax
                                    ;mov    al,[cs:drivenumber]     ; pass drive number to DBLSPACE as if
                                    ;mov    [cs:devdrivenum],al     ; it is a normal block device driver
0000171F C706[7D03]0000             mov     word [break_addr],0
00001725 A1[3724]                   mov     ax,[DevLoadEnd]
00001728 A3[7F03]                   mov     [break_addr+2],ax
0000172B A0[8503]                   mov     al,[drivenumber]        ; pass drive number to DBLSPACE as if
0000172E A2[8503]                   mov     [devdrivenum],al        ; it is a normal block device driver

00001731 B80A00                     mov     ax,10                   ; DS_INTERNAL_REVISION
                                                                    ; tell it what revision we expect
```

```
                                    ;call   far [cs:MagicBackdoor]; first time call is init entry point
00001734 FF1E[9003]                 call    far [MagicBackdoor]
                                                                ; with a standard device driver
                                                                ; init packet at es:bx
00001738 731D                       jnb     short no_driver_version_fail ; skip if not a version failure
0000173A B80600                     mov     ax,6                ; DS_INTERNAL_REVISION_6 ; (Stacker ?)
                                                                ; tell it what revision we expect
                                    ;call   far [cs:MagicBackdoor]
0000173D FF1E[9003]                 call    far [MagicBackdoor]
00001741 7314                       jnb     short no_driver_version_fail

                                ;   In this case, we're going to display a message

                                    ;push   cs
                                    ;pop    ds
                                    ; 13/04/2024
                                    ; ds = cs
00001743 BA[DA53]                   mov     dx,baddblspace      ; "Required system component is not instal"...
00001746 E80B33                     call    print               ; display the message

                                ;   point backdoor call back to safe far return

                                fail_driver_load:
                                    ;mov    [cs:MagicBackdoor+2],cs
                                    ;mov    word [cs:MagicBackdoor],NullBackdoor
00001749 8C0E[9203]                 mov     [MagicBackdoor+2],cs
0000174D C706[9003][9403]           mov     word [MagicBackdoor],NullBackdoor
                                pre_exit_err:
00001753 B84000                     mov     ax,40h              ; SYSPRE_BADFILE_ERROR
                                                                ; (problem loading dblspace.bin)
00001756 C3                         retn

                                no_driver_version_fail:
00001757 09C0                       or      ax,ax               ; error code returned?
00001759 75EE                       jnz     short fail_driver_load

                                magic_is_resident:
                                    ; 13/04/2024
                                    ; ds = cs
                                    ;mov    ax,[cs:break_addr]
0000175B A1[7D03]                   mov     ax,[break_addr]
0000175E E8B4FB                     call    ParaRound           ; convert to paragraphs
                                    ;add    ax,[cs:break_addr+2]
                                    ;mov    [cs:DevBrkAddr+2],ax
                                    ;mov    word [cs:DevBrkAddr],0; store normalized end here
00001761 0306[7F03]                 add     ax,[break_addr+2]
00001765 A3[3F24]                   mov     [DevBrkAddr+2],ax
00001768 C706[3D24]0000             mov     word [DevBrkAddr],0
0000176E BB0400                     mov     bx,4                ; inquire how many paragraphs it wants
                                    ;call   far [cs:MagicBackdoor]
00001771 FF1E[9003]                 call    far [MagicBackdoor]
                                    ;mov    bx,[cs:ALLOCLIM]    ; get top of free memory
00001775 8B1E[A502]                 mov     bx,[ALLOCLIM]
00001779 29C3                       sub     bx,ax               ; see how much we'll lower it
                                    ;cmp    bx,[cs:DevBrkAddr+2]  ; is there that much room free?
0000177B 3B1E[3F24]                 cmp     bx,[DevBrkAddr+2]
0000177F 7212                       jb      short cant_move_driver
                                    ;;sub   [cs:ALLOCLIM],ax    ; (mov [cs:ALLOCLIM],bx)
                                    ;mov    [cs:ALLOCLIM],bx ; Retro DOS v5.0 ; 08/04/2024
                                    ; 13/04/2024
00001781 891E[A502]                 mov     [ALLOCLIM],bx
                                    ;mov    es,[cs:ALLOCLIM]
00001785 8E06[A502]                 mov     es,[ALLOCLIM]
00001789 BB0600                     mov     bx,6                ; tell the driver to move itself
                                    ;call   far [cs:MagicBackdoor]
0000178C FF1E[9003]                 call    far [MagicBackdoor]
                                    ;mov    [cs:DevBrkAddr+2],ax  ; save end of low stub
00001790 A3[3F24]                   mov     [DevBrkAddr+2],ax     ; save end of low stub

                                cant_move_driver:
                                    ;mov    ax,[cs:DevBrkAddr+2]  ; get terminate segment
00001793 A1[3F24]                   mov     ax,[DevBrkAddr+2]
                                    ;cmp    ax,[cs:DevLoadEnd]    ; terminate size TOO big?
00001796 3B06[3724]                 cmp     ax,[DevLoadEnd]
0000179A 77B7                       ja      short pre_exit_err    ; error out if so

                                ;------ deal with block device drivers

                                _isblock:                       ; if no units found,erase the device
                                    ; 13/04/2024
                                    ; ds = cs
                                    ;mov    al,[cs:unitcount]
0000179C A0[7C03]                   mov     al,[unitcount]
0000179F 08C0                       or      al,al
000017A1 74B0                       jz      short pre_exit_err
000017A3 30E4                       xor     ah,ah
                                    ;lds    si,[cs:DevEntry]      ; set ds:si to header
000017A5 C536[3924]                 lds     si,[DevEntry]
000017A9 88440A                     mov     [si+10],al            ; mov [si+SYSDEV.NAME],al
                                                                  ; number of units in name field
                                                                  ; device drivers are *supposed*
                                                                  ; to do this for themselves.
000017AC 89C1                       mov     cx,ax
000017AE 2EC43E[6D02]               les     di,[cs:DOSINFO]             ; es:di point to dos info
000017B3 268A6520                   mov     ah,[es:di+20h]        ; [es:di+SYSI_NUMIO]
                                                                  ; get number of devices
000017B7 88E2                       mov     dl,ah
000017B9 00C4                       add     ah,al                 ; check for too many devices
000017BB 80FC1A                     cmp     ah,26                 ; 'A' - 'Z' is 26 devices
000017BE 7793                       ja      short pre_exit_err
000017C0 2E800E[6919]02             or      byte [cs:setdevmarkflag],2
000017C6 E83D1F                     call    DevSetBreak
                                    ;jnc    short _ok_block
                                    ;jmp    pre_exit_err
                                    ; 13/04/2024
000017C9 7288                       jc      short pre_exit_err  ; ds <> cs

                                _ok_block:
000017CB 26886520                   mov     [es:di+20h],ah        ; [es:di+SYSI_NUMIO] ; update the amount

000017CF 2EC51E[8103]               lds     bx,[cs:bpb_addr]      ; point to bpb array (*)
000017D4 30F6                       xor     dh,dh

                                _perunit:
000017D6 2EC42E[6D02]               les     bp,[cs:DOSINFO]
000017DB 26C46E00                   les     bp,[es:bp+0]          ; [es:bp.sysi_dpb]
                                                                  ; get first dpb
                                                                  ; [es:bp+SysInitvars.SYSI_DPB] ; [es:bp+0]
                                _scandpb:
000017DF 26837E19FF                 cmp     word [es:bp+19h],0FFFFh ; -1 ; [es:bp.dpb_next_dpb]
000017E4 7406                       jz      short _foundpb
000017E6 26C46E19                   les     bp,[es:bp+19h]        ; les bp,[es:bp.dpb_next_dpb]
                                                                  ; [es:bp+DPB.NEXT_DPB]
000017EA EBF3                       jmp     short _scandpb
```

```
29613
29614                                   ;   We've found the end of the DPB chain. Now extend it.
29615
29616                                   _foundpb:
29617 000017EC 2EA1[3D24]                   mov     ax,[cs:DevBrkAddr]
29618 000017F0 26894619                     mov     [es:bp+19h],ax          ; [es:bp.dpb_next_dpb] ; DPB.NEXT_DPB
29619 000017F4 2EA1[3F24]                   mov     ax,[cs:DevBrkAddr+2]
29620 000017F8 2689461B                     mov     [es:bp+1Bh],ax          ; [es:bp.dpb_next_dpb+2] ; DPB.NEXT_DPB+2
29621 000017FC 2EC42E[3D24]                 les     bp,[cs:DevBrkAddr]
29622 00001801 26C74619FFFF                 mov     word [es:bp+19h],0FFFFh ; -1
29623 00001807 26C64618FF                   mov     byte [es:bp+18h],0FFh ; [es:bp.dpb_first_access],-1
29624                                                                       ; DPB.FIRST_ACCESS
29625 0000180C 2E8306[3D24]3D               add     word [cs:DevBrkAddr],61 ; DPBSIZ ; 3Dh
29626 00001812 E8D01E                       call    RoundBreakAddr
29627 00001815 8B37                         mov     si,[bx]                 ; ds:si points to bpb (*)
29628                                                                       ; (mov si,[bx] ..and then.. add bx,2)
29629                                                                       ; Note: If unit count > 1,bx points to a BPB in the BPB array,
29630                                                                       ; the array address is in [bpb_addr] (*)
29631                                                                       ; Erdogan Tan - 07/07/2023
29632 00001817 26885600                     mov     [es:bp+0],dl                    ; mov word [es:bp.dpb_drive],dx
29633                                                                       ; [es:bp+DPB.DRIVE],dl
29634 0000181B 26887601                     mov     [es:bp+1],dh                    ; [es:bp+DPB.UNIT],dh
29635 0000181F 52                           push    dx
29636 00001820 51                           push    cx
29637 00001821 BA5241                       mov     dx,4152h                ; DX = signature 4152h ('AR') for FAT32 extended BPB/DPB
29638 00001824 31C9                         xor     cx,cx   ; 0
29639 00001826 26894E1D                     mov     [es:bp+1Dh],cx          ; DPB.NEXT_FREE ; last allocated cluster #
29640 0000182A 394C0B                       cmp     [si+0Bh],cx             ; BPB.fatsecs16 ; [si+A_BPB.BPB_SECTORSPERFAT]
29641 0000182D 7514                         jnz     short _setdpb           ; FAT DPB (33 bytes)
29642                                                                       ; FAT32 DPB (61 bytes)
29643 0000182F 26894E39                     mov     [es:bp+39h],cx          ; DPB.FAT32_NXTFREE = 0
29644 00001833 26894E3B                     mov     [es:bp+3Bh],cx          ; DPB.FAT32_NXTFREE+2 = 0
29645 00001837 49                           dec     cx                      ; 0FFFFh ; -1
29646 00001838 26894E1F                     mov     [es:bp+1Fh],cx          ; DPB.FREE_CNT (-1 = unknown)
29647 0000183C 26894E21                     mov     [es:bp+21h],cx          ; DPB.FREE_CNT+2 (-1 = unknown)
29648 00001840 B95845                       mov     cx,4558h                ; CX = signature 4558h ('EX') for FAT32 extended BPB/DPB
29649
29650                                   _setdpb:
29651 00001843 B453                         mov     ah,53h          ; SETDPB ; hidden system call
29652 00001845 CD21                         int     21h             ; DOS - 2+ internal - TRANSLATE BIOS PARAMETER BLOCK
29653                                                               ; DS:SI -> BPB (BIOS Parameter Block)
29654                                                               ; ES:BP -> buffer for DOS Drive Parameter Block
29655                                                               ; (if CX=4558h & DX=4152h,FAT32 Extended DPB will be set)
29656 00001847 59                           pop     cx
29657 00001848 5A                           pop     dx
29658 00001849 268B4602                     mov     ax,[es:bp+2]            ; [es:bp.dpb_sector_size]  ; [es:bp+DPB.SECTOR_SIZE]
29659 0000184D 06                           push    es
29660 0000184E 2EC43E[6D02]                 les     di,[cs:DOSINFO]
29661 00001853 263B4510                     cmp     ax,[es:di+10h]          ; [es:di.sysi_maxsec] ; [es:di+SysInitvars.SYSI_MAXSEC]
29662 00001857 07                           pop     es
29663 00001858 7604                         jbe     short _iblk_1
29664                                       ; 13/04/2024
29665                                       ; ds <> cs
29666 0000185A B84000                       mov     ax,40h                  ; SYSPRE_BADFILE_ERROR ; (pre_exit_err)
29667                                                                       ; (problem loading dblspace.bin)
29668 0000185D C3                           retn
29669
29670                                   _iblk_1:
29671 0000185E 1E                           push    ds
29672 0000185F 2EC506[3924]                 lds     ax,[cs:DevEntry]
29673 00001864 26894613                     mov     [es:bp+13h],ax          ; [es:bp+DPB.DRIVER_ADDR]
29674 00001868 268C5E15                     mov     [es:bp+15h],ds          ; [es:bp+DPB.DRIVER_ADDR+2]
29675 0000186C 1F                           pop     ds
29676 0000186D FEC2                         inc     dl                      ; increment drive number
29677 0000186F FEC6                         inc     dh                      ; increment unit number
29678 00001871 43                           inc     bx
29679 00001872 43                           inc     bx                      ; point to next BPB
29680                                                                       ; (in the BPB array) (*) -add bx,2-
29681 00001873 49                           dec     cx                      ; loop _foundpb
29682 00001874 7403                         jz      short _linkit
29683 00001876 E973FF                       jmp     _foundpb
29684
29685                                   _linkit:
29686 00001879 0E                           push    cs
29687 0000187A 1F                           pop     ds
29688 0000187B E825F5                       call    TempCDS                 ; set cds for new drives
29689                                       ; 13/04/2024
29690                                       ; (DS may not be same with CS here)
29691 0000187E 2EC43E[6D02]                 les     di,[cs:DOSINFO]                 ; es:di = dos table (SysInitVars)
29692 00001883 268B4522                     mov     ax,[es:di+22h]          ; [es:di+SYSI_DEV] ; dx:cx = head of list
29693 00001887 268B5D24                     mov     bx,[es:di+24h]          ; [es:di+SYSI_DEV+2]
29694 0000188B 2EC536[3924]                 lds     si,[cs:DevEntry]        ; ds:si = device location
29695 00001890 8904                         mov     [si],ax                 ; link in the driver
29696 00001892 895C02                       mov     [si+2],bx
29697 00001895 26897522                     mov     [es:di+22h],si          ; [es:di+SYSI_DEV] ; set head of list in dos
29698 00001899 268C5D24                     mov     [es:di+24h],ds          ; [es:di+SYSI_DEV+2]
29699 0000189D E8881E                       call    DevBreak                ; mark successful install
29700                                       ; 13/04/2024
29701                                       ; ds = cs
29702                                       ;mov    cx,[cs:DevBrkAddr+2]  ; pass it a work buffer
29703                                       ;mov    dx,[cs:ALLOCLIM]      ; address in cx (segment)
29704 000018A0 8B0E[3F24]                   mov     cx,[DevBrkAddr+2]
29705 000018A4 8B16[A502]                   mov     dx,[ALLOCLIM]
29706 000018A8 29CA                         sub     dx,cx                   ; for len dx (paragraphs)
29707 000018AA B80055                       mov     ax,5500h                ; we're shuffle aware,but don't move
29708                                                                       ; any drives at this point.
29709 000018AD BB0200                       mov     bx,2                    ; switch what we can now
29710                                       ;call   far [cs:MagicBackdoor]
29711 000018B0 FF1E[9003]                   call    far [MagicBackdoor]
29712                                   pre_exit:                           ; no errors!
29713 000018B4 31C0                         xor     ax,ax                   ; zf=1
29714                                   no_magic:   ; 13/04/2024
29715 000018B6 C3                           retn
29716
29717                                   ; =============== S U B R O U T I N E =======================================
29718
29719                                   ; 08/04/2024 - Retro DOS v5.0
29720                                   ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:1B9Fh)
29721
29722                                   ;***MagicPostload -- called to clean up and make sure Magic is final placed
29723
29724                                   MagicPostload:
29725                                       ; 13/04/3024
29726                                       ; ds = cs
29727 000018B7 E89C00                     call    get_dblspace_version  ; is it there?
29728 000018BA 75FA                       jnz     short no_magic        ; done if not
29729 000018BC F7C20080                   test    dx,8000h              ; is it already permanent?
29730 000018C0 74F4                       jz      short no_magic        ; no,done if so (not in final position)
29731 000018C2 BBFFFF                     mov     bx,0FFFFh ; -1        ; how much space does it want?
29732 000018C5 B8114A                     mov     ax,4A11h              ; multMagicdrv
29733                                                                   ; DBLSPACE.BIN - GET RELOCATION SIZE
29734 000018C8 CD2F                       int     2Fh                   ; get paragraphs into ax
29735 000018CA 40                         inc     ax                    ; extra 2 paragraphs for the stub
29736 000018CB 40                         inc     ax                    ; ((tiny_stub_end-tiny_stub_start)+15)/16
```

```
                                                    ; (18+15)/16 = 2
                                ;mov    [cs:DevSize],ax             ; store that (**)
                                ;mov    byte [cs:DeviceHi],0  ; not to be loaded in UMB
                                ;mov    [cs:bpb_addr+2],cs    ; pass name so that
                                ;                             ; arena header can be set
                                ;mov    word [cs:bpb_addr],MagicDDName ; "\DBLSPACE.BIN"
                                ; 13/04/2024
                                ; ds = cs
000018CC A3[3324]               mov     [DevSize],ax
000018CF C606[5124]00           mov     byte [DeviceHi],0
000018D4 8C0E[8303]             mov     [bpb_addr+2],cs
000018D8 C706[8103][AB16]       mov     word [bpb_addr],MagicDDName

000018DE E8112F                 call    round                 ; normalize memhi:memlo
000018E1 E8811C                 call    InitDevLoad           ; set up sub-arena,DevLoadAddr,
                                                              ; DevLoadEnd,and DevEntry
                                                              ; gets arena name from bpb_addr
                                ; 13/04/2024
                                ; ds = cs
                                ;mov    es,[cs:DevLoadAddr]   ; (**) (InitDevload sets this)
000018E4 8E06[3524]             mov     es,[DevLoadAddr]

                              ;   First, move a little header in place so that this looks
                              ;   to the mem command like a legitimate driver load. Otherwise,
                              ;   it will display garbage for the device name

000018E8 31FF                   xor     di,di                 ; move a little header in place
                                                              ; so that this looks to the mem command
                                                              ; like a legitimate driver load
000018EA BE[C816]               mov     si,tiny_stub_start
                                ;mov    cx,18                 ; (tiny_stub_end-tiny_stub_start)
000018ED B91200                 mov     cx,tiny_stub_end-tiny_stub_start
000018F0 F3A4                   rep movsb                     ; move it!
000018F2 8CC0                   mov     ax,es                 ; advance es appropriately
000018F4 40                     inc     ax                    ; add ax,((tiny_stub_end-tiny_stub_start)+15)/16
000018F5 40                     inc     ax
000018F6 8EC0                   mov     es,ax
000018F8 BBFEFF                 mov     bx,0FFFEh ; -2        ; final placement!
000018FB B8114A                 mov     ax,4A11h              ; multMagicdrv
000018FE CD2F                   int     2Fh                   ; DBLSPACE.BIN - RELOCATE
                                                              ; es = segment to which to relocate DBLSPACE.BIN
                                ;mov    ax,[cs:DevLoadAddr] ; (**)
                                ;add    ax,[cs:DevSize]             ; calculate seg after DD load
                                ;mov    [cs:DevBrkAddr+2],ax  ; save as ending address!
                                ;mov    word [cs:DevBrkAddr],0
                                ; 13/04/2024
                                ; ds = cs
00001900 A1[3524]               mov     ax,[DevLoadAddr]
00001903 0306[3324]             add     ax,[DevSize]
00001907 A3[3F24]               mov     [DevBrkAddr+2],ax
0000190A C706[3D24]0000         mov     word [DevBrkAddr],0

00001910 E8F31D                 call    DevSetBreak           ; go ahead and alloc mem for device
                                ;call   DevBreak
                              ;no_magic:
                                ;retn
                                ; 13/04/2024 - Retro DOS v5.0
00001913 E9121E                 jmp     DevBreak

                              ; =============== S U B R O U T I N E ===================================

                              ; 08/04/2024 - Retro DOS v5.0
                              ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:1C08h)

                              ;***MagicSetCdss -- disable CDSs for still unmounted DblSpace drives
                              ;
                              ;   entry:
                              ;      CDSs are now persistent and in their final place

                              MagicSetCdss:
                                ; 13/04/2024 - Retro DOS v5.0
                                ; ds = cs
00001916 E83D00                 call    get_dblspace_version  ; is it there?
00001919 753A                   jnz     short magic_set_exit  ; done if not
                                                              ; cl = first DblSpace drive in ASCII
                                                              ; ch = number of DblSpace drive letters
0000191B 2EC536[6D02]           lds     si,[cs:DOSINFO]             ; point to DOS data area (SysInitVars)
00001920 C57416                 lds     si,[si+16h]           ; lds si,[si+SYSI_CDS] ; fetch CDSs
00001923 B458                   mov     ah,88                 ; curdirLen
00001925 80E941                 sub     cl,'A'                ; make it zero based.
00001928 88C8                   mov     al,cl                 ; get first DblSpace drive letter
0000192A F6E4                   mul     ah                    ; find first DblSpace CDS
0000192C 01C6                   add     si,ax                 ; cds pointer
0000192E 88CA                   mov     dl,cl                 ; save for drive testing loop
00001930 88E9                   mov     cl,ch                 ; get DblSpace drive count into cx
00001932 30ED                   xor     ch,ch

                              ;   We know cx > 0, or else the driver wouldn't have stayed resident

                              magic_set_cdss_1:
00001934 51                       push    cx
00001935 52                       push    dx
00001936 1E                       push    ds
00001937 56                       push    si
00001938 B8114A                   mov     ax,4A11h              ; multMagicdrv
0000193B BB0100                   mov     bx,1                  ; MD_DRIVE_MAP ; inquire drive map
0000193E CD2F                     int     2Fh                   ; DBLSPACE.BIN - "GetDriveMapping"
                                                                ; see if this is an unused DblSpace drive
00001940 5E                       pop     si
00001941 1F                       pop     ds
00001942 5A                       pop     dx
00001943 59                       pop     cx
00001944 38DA                     cmp     dl,bl                 ; if mapped to itself,it is vacant
00001946 7504                     jnz     short magic_set_cdss_2; skip if used
00001948 806444BF                 and     byte [si+44h],0BFh    ; Retro DOS v5.0 ; 08/04/2024
                                  ;and    word [si+43h],0BFFFh
                                                                ; reset the bit in flags (curdir_inuse bit)
                                                                ; [si+curdir_list.cdir_flags],~curdir_inuse ; word
                                                                ; (.. [si+1+curdir_list.cdir_flags],0BFh ; byte)
                              magic_set_cdss_2:
0000194C 83C658                   add     si,88                 ; curdirLen
0000194F FEC2                     inc     dl                    ; next drive
00001951 E2E1                     loop    magic_set_cdss_1
                                  ; 13/04/2024
00001953 0E                       push    cs
00001954 1F                       pop     ds
                              magic_set_exit:
00001955 C3                       retn

                              ; =============== S U B R O U T I N E ===================================

                              ; 08/04/2024 - Retro DOS v5.0
                              ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:1C47h)
```

```
                                get_dblspace_version:
00001956 B8114A                     mov     ax,4A11h        ; multMagicdrv
                                                            ; DBLSPACE.BIN - "GetVersion" - INSTALLATION CHECK
                                                            ; (BX = 0)
00001959 31DB                       xor     bx,bx           ; MD_VERSION = 0
0000195B CD2F                       int     2Fh             ; Return:
                                                            ;  AX = 0000h (successful)
                                                            ;  BX = 444Dh ("DM")
                                                            ;  CL = first drive letter used by DBLSPACE (41h = A:)
                                                            ;  CH = number of drive letters used by DBLSPACE
                                                            ;  DX = internal DBLSPACE.BIN version number (bits 14-0)
                                                            ;  bit 15 set if DBLSPACE.BIN has not yet been relocated
                                                            ;  to final position in memory (i.e. DBLSPACE.SYS /MOVE)
0000195D 09C0                       or      ax,ax           ; ax = 0 (successful,zf=1)
0000195F C3                         retn

                                ; ----------------------------------------------------------------------------
                                ; SYSCONF.ASM - MSDOS 6.0 - 1991
                                ; ----------------------------------------------------------------------------
                                ; 27/03/2019 - Retro DOS v4.0

                                ;MULTI_CONFIG      equ 1

                                HIGH_FIRST  equ 080h              ; from ARENA.INC - modifier for
                                                                        ; allocation strategy call

                                ;have_install_cmd equ 00000001b   ; config.sys has install= commands
                                ;has_installed       equ 00000010b       ; sysinit_base installed.

                                default_filenum equ 8

                                ;stacksw    equ true              ; include switchable hardware stacks

                                ; external variable defined in ibmbio module for multi-track

                                ;multrk_on  equ 10000000b         ;user spcified mutitrack=on,or system turns
                                                                  ; it on after handling config.sys file as a
                                                                  ; default value,if multrk_flag = multrk_off1.
                                ;multrk_off1       equ 00000000b         ;initial value. no "multitrack=" command entered.
                                ;multrk_off2       equ 00000001b         ;user specified multitrack=off.

                                ; if stacksw

                                ; internal stack parameters

                                ;entrysize  equ 8

                                ;mincount   equ 8
                                ;defaultcount      equ 9
                                ;maxcount   equ 64

                                ;minsize    equ 32
                                ;defaultsize       equ 128
                                ;maxsize    equ 512

                                DOS_FLAG_OFFSET    equ 86h

                                ;ifdef MULTI_CONFIG
                                ;
                                ;   config_envlen must immediately precede config_wrkseg, because they
                                ;   may be loaded as a dword ptr

                                ; 30/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; 25/10/2022
00001960 0000                   config_envlen:     dw  0                ; when config_wrkseg is being used as
                                                                        ;  a scratch env, this is its length
00001962 0000                   config_wrkseg:     dw  0                ; config work area (above confbot)
                                                                        ;  segment of work area

00001964 00                     config_cmd: db  0                ; current config cmd
                                                                 ;  (with CONFIG_OPTION_QUERY bit intact)
00001965 00                     config_multi:      db  0                 ; non-zero if multi-config config.sys

                                ;endif ; MULTI_CONFIG

00001966 00                     multdeviceflag:    db  0

00001967 0000                   devmark_addr:      dw  0                ;segment address for devmark.

00001969 00                     setdevmarkflag: db  0                   ;flag used for devmark

                                ; 30/12/2022
                                ; 12/12/2022
0000196A 00                     driver_units:      db  0                ;total unitcount for driver

                                ; 12/12/2022
                                ;ems_stub_installed:
                                ;          db  0

                                ; 12/12/2022
                                ;align 2

                                badparm_ptr:       ; label dword
0000196B 0000                   badparm_off:       dw  0
0000196D 0000                   badparm_seg:       dw  0

                                ;****************************************************************************
                                ;take care of config.sys file.
                                ;system parser data and code.
                                ;****************************************************************************

                                ;*********************************************************************
                                ; parser options set for msbio sysconf module
                                ;*********************************************************************
                                ;
                                ;**** default assemble swiches definition ***************************

                                ;farsw      equ 0          ; near call expected
                                ;datesw     equ 0          ; check date format
                                ;timesw     equ 0          ; check time format
                                ;filesw     equ 1          ; check file specification
                                ;capsw      equ 0          ; perform caps if specified
                                ;cmpxsw     equ 0          ; check complex list
                                ;numsw      equ 1          ; check numeric value
                                ;keysw      equ 0          ; support keywords
                                ;swsw       equ 1          ; support switches
                                ;val1sw     equ 1          ; support value definition 1
                                ;val2sw     equ 0          ; support value definition 2
                                ;val3sw     equ 1          ; support value definition 3
                                ;drvsw      equ 1          ; support drive only format
                                ;qussw      equ 0          ; support quoted string format

                                ; psdata_seg equ cs
```

```
                            ;.xlist
                            ;include parse.asm          ;together with psdata.inc
                            ;.list

                        ; PSDATA.INC - MSDOS 6.0 - 1991
                        ; ============================================================================
                        ; 27/03/2019 - Retro DOS v4.0

                        ; 30/03/2019
                        ; VERSION.INC (MSDOS 6.0)
                        ; Set DBCS Blank constant

                        ; ifndef DBCS
                        DB_SPACE EQU 2020h
                        DB_SP_HI EQU 20h
                        DB_SP_LO EQU 20h
                        ; else

                        ;****************************************************************************
                        ; Parser include file
                        ;****************************************************************************

                        ;**** Equation field
                        ;-------- Character code definition

                        _$P_DBSP1       equ DB_SP_HI        ;AN000; 1st byte of DBCS blank
                        _$P_DBSP2       equ DB_SP_LO        ;AN000; 2nd byte of DBCS blank
                        _$P_Period      equ "."              ;AN020;
                        _$P_Slash       equ "/"              ;AN020;
                        _$P_Space       equ " "              ;AN000; SBCS blank
                        _$P_Comma       equ ","              ;AN000;
                        _$P_Switch      equ "/"              ;AN000;
                        _$P_Keyword     equ "="              ;AN000;
                        _$P_Colon       equ ":"              ;AN000;
                        _$P_Plus        equ "+"              ;AN000;
                        _$P_Minus       equ "-"              ;AN000;
                        _$P_Rparen      equ ")"              ;AN000;
                        _$P_Lparen      equ "("              ;AN000;
                        ;_$P_SQuote         equ  "'"                    ;AN025; deleted
                        _$P_DQuote      equ '"'              ;AN000;
                        _$P_NULL        equ 0               ;AN000;
                        _$P_TAB                 equ 9               ;AN000;
                        _$P_CR                  equ 0Dh             ;AN000;
                        _$P_LF                  equ 0Ah             ;AN000;
                        _$P_ASCII80     equ 80h             ;AN000; ASCII 80h character code

                        ;-------- Masks
                        _$P_Make_Lower          equ 20h             ;AN000; make lower case character
                        _$P_Make_Upper          equ 0FFh-_$P_Make_Lower ;AN000; make upper case character

                        ;-------- DOS function call related equs

                        _$P_DOS_Get_CDI         equ 3800h           ;AN000; get country dependent information
                                                                    ; by this call, following information
                        struc _$P_CDI
                         .DateF: resw 1
                         .Money: resb 5
                         .1000:      resb 2
                         .Dec:       resb 2
                         .DateS: resb 2
                         .TimeS: resb 2
                             resb 1
                             resb 1
                         .TimeF: resb 1
                             resw 2
                             resb 2
                             resw 5
                         .size:
                        endstruc

                        _$P_Date_MDY            equ 0               ;AN000;
                        _$P_Date_DMY            equ 1               ;AN000;
                        _$P_Date_YMD            equ 2               ;AN000;
                        ;-------------
                        _$P_DOS_GetEV           equ 6300h           ;AN000; get DBCS EV call
                                                                    ;AN000; DS:SI will points to DBCS EV
                        ;-------------
                        _$P_DOS_Get_TBL         equ 65h             ;AN000; get uppercase table call
                                                                    ;AN000; following parameters are set
                                                                    ;AN000; to get casemap table.
                        _$P_DOSTBL_Def          equ -1              ;AN000; get default
                        _$P_DOSTBL_BL           equ 5               ;AN000; buffer length for Tbl pointer
                        _$P_DOSTBL_File         equ 4               ;AN000; get file uppercase table
                        _$P_DOSTBL_Char         equ 2               ;AN000; get character uppercase table
                                                                    ; By this call following information
                                                                    ; is returned.
                        struc _$P_DOS_TBL
                         .InfoID: resb 1                ;AN000; information id for the table
                         .Off:       resw 1                      ;AN000; offset address of the table
                         .Seg:       resw 1                      ;AN000; segment address of the table
                        endstruc

                        ; ---------------------------------------------------------------------------
                        ; PARMS     LABEL   BYTE
                        ;           DW      PARMSX
                        ;           DB      2               ; NUMBER OF STRINGS (0, 1, 2)
                        ;           DB      length          ; LENGTH OF THE NEXT LIST, 0 IF NONE
                        ;           DB      " .. "           ; EXTRA DELIMITER LIST,
                        ;                                   ; TYPICAL ARE ";", "="
                        ;                                   ; "," & WHITESPACE ALWAYS
                        ;           DB      length          ; LENGTH OF THE NEXT LIST, 0 IF NONE
                        ;           DB      " .. "           ; EXTRA END OF LINE LIST, CR, LF OR 0 ALWAYS
                        ; ---------------------------------------------------------------------------

                        ;-------------------------------- PARMS block structure
                        struc _$P_PARMS_Blk
                         .PARMSX_Address:  resw 1           ;AN000; Address of PARMSX
                         .Num_Extra:          resb 1                ;AN000; Number of extra stuff
                         .Len_Extra_Delim: resb 1           ;AN000; Length of extra delimiter
                        endstruc

                        _$P_Len_PARMS           equ 4               ;AN000;
                        _$P_I_Use_Default  equ    0                 ;AN000; no extra stuff specified
                        _$P_I_Have_Delim   equ    1                 ;AN000; extra delimiter specified
                        _$P_I_Have_EOL          equ 2               ;AN000; extra EOL specified

                        ; ---------------------------------------------------------------------------
                        ; PARMSX    LABEL   BYTE
                        ;           DB      minp,maxp       ; MIN, MAX POSITIONAL OPERANDS ALLOWED
                        ;           DW      CONTROL         ; DESCRIPTION OF POSITIONAL 1
                        ;           :                       ; REPEATS maxp-1 TIMES
                        ;           DB      maxs            ; # OF SWITCHES
                        ;           DW      CONTROL         ; DESCRIPTION OF SWITCH 1
                        ;           :                       ; REPEATS maxs-1 TIMES
```

```
30109                           ;           DB      maxk            ; # OF KEYWORD
30110                           ;           DW      CONTROL         ; DESCRIPTION OF KEYWORD 1
30111                           ;           :                       ; REPEATS maxk-1 TIMES
30112                           ; ---------------------------------------------------------------------------
30113
30114                           ;-------------------------------- PARMSX block structure
30115                           struc _$P_PARMSX_Blk                ;AN000;
30116 00000000 ??                 .MinP: resb 1                    ;AN000; Minimum positional number
30117 00000001 ??                 .MaxP:     resb 1                ;AN000; Maximum positional number
30118 00000002 ????               .1st_Control: resw 1             ;AN000; Address of the 1st CONTROL block
30119                           endstruc
30120
30121                           ; ---------------------------------------------------------------------------
30122                           ; << Control field definition  >>
30123                           ;
30124                           ;
30125                           ;CONTROL   LABEL   BYTE
30126                           ;       DW      MATCH_FLAGS     ; CONTROLS TYPE MATCHED
30127                           ;                               ; 8000H=NUMERIC VALUE, (VALUE LIST WILL BE CHECKED)
30128                           ;                               ; 4000H=SIGNED NUMERIC VALUE (VALUE LIST WILL BE CHECKED)
30129                           ;                               ; 2000H=SIMPLE STRING(VALUE LIST WILL BE CHECKED)
30130                           ;                               ; 1000H=DATE STRING (VALUE LIST WON'T BE CHECKED)
30131                           ;                               ; 0800H=TIME STRING (VALUE LIST WON'T BE CHECKED)
30132                           ;                               ; 0400H=COMPLEX LIST (VALUE LIST WON'T BE CHECKED)
30133                           ;                               ; 0200H=FILE SPEC (VALUE LIST WON'T BE CHECKED)
30134                           ;                               ; 0100H=DRIVE ONLY (VALUE LIST WON'T BE CHECKED)
30135                           ;                               ; 0080H=QUOTED STRING (VALUE LIST WON'T BE CHECKED)
30136                           ;                               ; 0010H=IGNORE ":" AT END IN MATCH
30137                           ;                               ; 0002H=REPEATS ALLOWED
30138                           ;                               ; 0001H=OPTIONAL
30139                           ;       DW      FUNCTION_FLAGS
30140                           ;                               ; 0001H=CAP RESULT BY FILE TABLE
30141                           ;                               ; 0002H=CAP RESULT BY CHAR TABLE
30142                           ;                               ; 0010H=REMOVE ":" AT END
30143                           ; (tm10)                        ; 0020H=colon is not necessary for switch
30144                           ;       DW      RESULT          ; RESULT BUFFER
30145                           ;       DW      VALUES          ; VALUE LISTS
30146                           ;       DB      nid             ; NUMBER OF KEYWORD/SWITCH SYNONYMS IN FOLLOWING LIST
30147                           ;       DB      "...",0         ; IF n >0, KEYWORD 1
30148                           ;       :
30149                           ;
30150                           ;Note:
30151                           ;     - The MATCH_FLAG is bit significant. You can set, for example, TIME bit and
30152                           ;       DATE bit simalteniously.
30153                           ;
30154                           ;       The parser examins each bit along with the following priority.
30155                           ;
30156                           ;       COMPLEX -> DATE -> TIME -> NUMERIC VAL -> SIGNED NUMERIC VAL -> DRIVE ->
30157                           ;       FILE SPEC -> SIMPLE STRING.
30158                           ;
30159                           ;     - When the FUNCTION_FLAG is 0001 or 0002, the STRING pointed to by a pointer
30160                           ;       in the result buffer is capitalized.
30161                           ;
30162                           ;     - Match_Flags 0001H and 0002H have meaning only for the positional.
30163                           ;
30164                           ;     - The "...",0 (bottom most line) does require '=' or '/'. When you need a
30165                           ;       switch, for example, '/A', then STRING points to;
30166                           ;
30167                           ;                   DB    1         ; number of following synonyms
30168                           ;                   DB   '/A',0
30169                           ;
30170                           ;       When you need a keyword, for example, 'CODEPAGE=', then "...",0 will be;
30171                           ;
30172                           ;                   DB    1         ; number of following synonyms
30173                           ;                   DB   'CODEPAGE=',0
30174                           ;
30175                           ;     - "..." must consist of upper case characters only because the parser
30176                           ;       performs pattern matching after converting input to upper case (by
30177                           ;       using the current country upper case table)
30178                           ;
30179                           ;     - One "..." can contain only one switch or keyword. If you need, for
30180                           ;       example /A and /B, the format will be;
30181                           ;
30182                           ;                   DB    2         ; number of following synonyms
30183                           ;                   DB    '/A',0
30184                           ;                   DB    '/B',0
30185                           ; ---------------------------------------------------------------------------
30186
30187                           ;**** Match_Flags
30188
30189                           _$P_Num_Val    equ  8000h           ;AN000; Numeric Value
30190                           _$P_SNum_Val          equ  4000h    ;AN000; Signed numeric value
30191                           _$P_Simple_S          equ  2000h    ;AN000; Simple string
30192                           _$P_Date_S     equ  1000h           ;AN000; Date string
30193                           _$P_Time_S     equ  0800h           ;AN000; Time string
30194                           _$P_Cmpx_S     equ  0400h           ;AN000; Complex string
30195                           _$P_File_Spc          equ  0200h    ;AN000; File Spec
30196                           _$P_Drv_Only          equ  0100h    ;AN000; Drive Only
30197                           _$P_Qu_String         equ  0080h    ;AN000; Quoted string
30198                           _$P_Ig_Colon          equ  0010h    ;AN000; Ignore colon at end in match
30199                           _$P_Repeat     equ  0002h           ;AN000; Repeat allowed
30200                           _$P_Optional          equ  0001h    ;AN000; Optional
30201
30202                           ;**** Function flags
30203
30204                           _$P_CAP_File          equ  0001h    ;AN000; CAP result by file table
30205                           _$P_CAP_Char          equ  0002h    ;AN000; CAP result by character table
30206                           _$P_Rm_Colon          equ  0010h    ;AN000; Remove ":" at the end
30207                           _$P_colon_is_not_necessary equ 0020h  ;AN000;(tm10) /+10 and /+:10
30208
30209                           ;-------------------------------- Control block structure
30210                           struc _$P_Control_Blk
30211 00000000 ????               .Match_Flag:       resw 1        ;AN000; Controls type matched
30212 00000002 ????               .Function_Flag: resw 1           ;AN000; Function should be taken
30213 00000004 ????               .Result_Buf:       resw 1        ; Result buffer address
30214 00000006 ????               .Value_List:       resw 1        ;AN000; Value list address
30215 00000008 ??                 .nid:              resb 1        ;AN000; # of keyword/SW synonyms
30216 00000009 ??                 .KEYorSW:   resb 1         ;AN000; keyword or sw
30217                           endstruc
30218
30219                           ; ---------------------------------------------------------------------------
30220                           ; << Value List Definition >>
30221                           ;
30222                           ;VALUES     LABEL   BYTE
30223                           ;           DB      nval                ; NUMBER OF VALUE DEFINITIONS (0 - 3)
30224                           ;         +-
30225                           ;         | DB      nrng                ; NUMBER OF RANGES
30226                           ;         | +DB     ITEM_TAG            ; RETURN VALUE IF RANGE MATCHED
30227                           ;         | +DD     X,Y                 ; RANGE OF VALUES
30228                           ;         | :
30229                           ;         | DB      nnval               ; NUMBER OF CHOICES
30230                           ;         | +DB     ITEM_TAG            ; RETURN VALUE IF NUMBER CHOICE MATCHED
30231                           ;         | +DD     VALUE               ; SPECIFIC CHOICE IF NUMBER
30232                           ;         | :
```

```
30233                                   ;        | DB     nstrval              ; NUMBER OF CHOICES
30234                                   ;        | +DB    ITEM_TAG             ; RETURN VALUE IF STRING CHOICE MATCHED
30235                                   ;        | +DW    STRING               ; SPECIFIC CHOICE IF STING
30236                                   ;        +- :
30237                                   ;
30238                                   ;STRING    DB     "...",0                ; ASCIIZ STRING IMAGE
30239                                   ;
30240                                   ;Note:
30241                                   ;    - ITEM_TAG must not be 0FFH, which will be used in the result buffer
30242                                   ;      when no choice lists are provided.
30243                                   ;
30244                                   ;    - STRING must consist of upper case characters only because the parser
30245                                   ;      performs pattern matching after converting input to upper case (by
30246                                   ;      using the current country upper case table)
30247                                   ; ---------------------------------------------------------------------------
30248
30249                                   _$P_nval_None         equ 0             ;AN000; no value list ID
30250                                   _$P_nval_Range        equ 1             ;AN000; range list ID
30251                                   _$P_nval_Value        equ 2             ;AN000; value list ID
30252                                   _$P_nval_String       equ 3             ;AN000; string list ID
30253                                   _$P_Len_Range         equ 9             ;AN000; Length of a range choice(two DD plus one DB)
30254                                   _$P_Len_Value         equ 5             ;AN000; Length of a value choice(one DD plus one DB)
30255                                   _$P_Len_String        equ 3             ;AN000; Length of a string choice(one DW plus one DB)
30256                                   _$P_No_nrng    equ 0             ;AN000; (tm07) no nrng. nnval must not be 0.
30257
30258                                   struc _$P_Val_List
30259 00000000 ??                        .NumofList: resb 1                    ;AN000; number of following choice
30260 00000001 ????                      .Val_XL:    resw 1                    ;AN000; lower word of value
30261 00000003 ????                      .Val_XH:    resw 1                    ;AN000; higher word of value
30262 00000005 ????                      .Val_YL:    resw 1                    ;AN000; lower word of another value
30263 00000007 ????                      .Val_YH:    resw 1                    ;AN000; higher word of another value
30264                                   endstruc
30265
30266                                   ; ---------------------------------------------------------------------------
30267                                   ; << Result Buffer Definition  >>
30268                                   ;
30269                                   ;RESULT    LABEL  BYTE                   ; BELOW FILLED IN FOR DEFAULTS
30270                                   ;          DB     type                   ; TYPE RETURNED: 0=RESERVED,
30271                                   ;                                        ;       1=NUMBER, 2=LIST INDEX,
30272                                   ;                                        ;       3=STRING, 4=COMPLEX,
30273                                   ;                                        ;       5=FILESPEC, 6=DRIVE
30274                                   ;                                        ;       7=DATE, 8=TIME
30275                                   ;                                        ;       9=QUOTED STRING
30276                                   ;          DB     ITEM_TAG               ; MATCHED ITEM TAG
30277                                   ;
30278                                   ;          dw     synonym@               ; es:@ points to found SYNONYM if provided.
30279                                   ;
30280                                   ;            +-
30281                                   ;        | DD     n                      ; VALUE IF NUMBER
30282                                   ;        | or
30283                                   ;        |  DW     i                      ; INDEX (OFFSET) INTO VALUE LIST
30284                                   ;        |                                ; (ES presents Segment address)
30285                                   ;        | or
30286                                   ;        |  DD     STRING                 ; OFFSET OF STRING VALUE
30287                                   ;        | or
30288                                   ;        |  DB     drv                    ; DRIVE NUMBER (1-A, 2-B,..., 26-Z)
30289                                   ;        | or
30290                                   ;        |  DW     YEAR       ;(1980-2099)  IN CASE OF DATE
30291                                   ;        |  DB     MONTH      ;(1-12)      Note: Range check is not performed.
30292                                   ;        |  DB     DATE       ;(1-31)            0 is filled when the corresponding field was not specified.
30293                                   ;        | or
30294                                   ;        |  DB     HOUR       ;(0-23)       IN CASE OF TIME
30295                                   ;        |  DB     MINUTES    ;(0-59)      Note: Range check is not performed .
30296                                   ;        |  DB     SECONDS    ;(0-59)            0 is filled when the corresponding field was not specified .
30297                                   ;        |  DB     HUNDREDTHS ;(0-99)
30298                                   ;       +-
30299                                   ;
30300                                   ;
30301                                   ;Note: ITEM_TAG is 0FFH when the caller does not specify the choice
30302                                   ;      list.
30303                                   ;
30304                                   ;      YEAR: If the input value for the year is less than 100, parser
30305                                   ;        adds 1900 to it.  For example, when 87 is input to parser for
30306                                   ;        the year value, he returns 1987.
30307                                   ; ---------------------------------------------------------------------------
30308
30309                                   ;-------------------------------- Result block structure
30310                                   struc _$P_Result_Blk
30311 00000000 ??                        .Type:        resb 1              ;AN000; Type returned
30312 00000001 ??                        .Item_Tag:    resb 1              ;AN000; Matched item tag
30313 00000002 ????                      .SYNONYM_Ptr: resw 1              ;AN000; pointer to Synonym list returned
30314 00000004 ????????                  .Picked_Val:  resb 4              ;AN000; value
30315                                   endstruc
30316
30317                                   ;--------------------------------
30318                                   ;**** values for the type field in the result block
30319
30320                                   _$P_EOL               equ 0               ;AN000; End of line
30321                                   _$P_Number     equ 1                  ;AN000; Number
30322                                   _$P_List_Idx          equ 2               ;AN000; List Index
30323                                   _$P_String     equ 3                  ;AN000; String
30324                                   _$P_Complex    equ 4                  ;AN000; Complex
30325                                   _$P_File_Spec         equ 5               ;AN000; File Spec
30326                                   _$P_Drive      equ 6                  ;AN000; Drive
30327                                   _$P_Date_F     equ 7                  ;AN000; Date
30328                                   _$P_Time_F     equ 8                  ;AN000; Time
30329                                   _$P_Quoted_String  equ    9               ;AN000; Quoted String
30330
30331                                   _$P_No_Tag     equ 0FFh               ;AN000; No ITEM_TAG found
30332
30333                                   ;**** Return code
30334                                   ;
30335                                   ; following return code will be returned in the AX register.
30336
30337                                   _$P_No_Error          equ 0               ;AN000; No error
30338                                   _$P_Too_Many          equ 1               ;AN000; Too many operands
30339                                   _$P_Op_Missing        equ 2               ;AN000; Required operand missing
30340                                   _$P_Not_In_SW         equ 3               ;AN000; Not in switch list provided
30341                                   _$P_Not_In_Key        equ 4               ;AN000; Not in keyword list provided
30342                                   _$P_Out_Of_Range   equ    6               ;AN000; Out of range specified
30343                                   _$P_Not_In_Val        equ 7               ;AN000; Not in value list provided
30344                                   _$P_Not_In_Str        equ 8               ;AN000; Not in string list provided
30345                                   _$P_Syntax     equ 9                  ;AN000; Syntax error
30346                                   _$P_RC_EOL     equ -1                 ;AN000; End of command line
30347
30348                                   ; DATA - Retro DOS v4.0 - 27/03/2019
30349
30350                                   ; MSDOS 6.2 IO.SYS SYSINIT:179Ch
30351
30352                                   ; 14/04/2024 - Retro DOS v5.0
30353                                   ; PCDOS 7.1 IBMBIO.COM - SYSINIT:1C62h
30354
30355                                   ;********************** Local Data ************************************
30356 0000196F 0000                    _$P_ORDINAL:          dw  0               ;AN000; Operand ordinal save area
```

```
30357 00001971 0000                   _$P_RC:               dw  0             ;AN000; Return code from parser
30358 00001973 0000                   _$P_SI_Save:          dw  0             ;AN000; Pointer of command buffer
30359 00001975 0000                   _$P_DX:               dw  0             ;AN000; Return result buffer address
30360 00001977 00                     _$P_Terminator:       db  0             ;AN000; Terminator code (ASCII)
30361 00001978 0000                   _$P_DBCSEV_OFF:       dw  0             ;AN000; Offset of DBCS EV
30362 0000197A 0000                   _$P_DBCSEV_SEG:       dw  0             ;AN000; Segment of DBCS EV
30363 0000197C 0000                   _$P_Flags:    dw  0             ;AN000; Parser internal flags
30364                                 %define _$P_Flags1 _$P_Flags            ;AN038; to reference first byte flags
30365                                 %define _$P_Flags2 _$P_Flags+1          ;AN038; to reference second byte flags only
30366
30367                                 ;in second byte of _$P_Flags, referenced as _$P_Flags2:
30368                                 _$P_equ               equ 01h          ;AN000; "=" packed in string buffet
30369                                 _$P_Neg               equ 02h          ;AN000; Negative value
30370                                 _$P_Time12    equ 04h          ;AN000; set when PM is specified
30371                                 _$P_Key_Cmp   equ 08h          ;AN000; set when keyword compare
30372                                 _$P_SW_Cmp    equ 10h          ;AN000; set when switch compare
30373                                 _$P_Extra     equ 20h          ;AN000; set when extra delimiter found
30374                                 _$P_SW                equ 40h          ;AN000; set when switch found (tm08)
30375                                 _$P_Signed    equ 80h          ;AN000; signed numeric specified
30376
30377                                 ;in first byte of _$P_Flags, referenced as _$P_Flags1:
30378                                 _$P_time12am          equ 01h          ;AN038; set when AM is specified on time
30379                                 _$P_TIME_AGAIN        equ 02h          ;AN039; SET WHEN READY TO RE-PARSE TIME
30380
30381 0000197E 0000                   _$P_SaveSI_Cmpx:   dw     0             ;AN000; save si for later use by complex
30382 00001980 0000                   _$P_KEYorSW_Ptr:   dw     0             ;AN000; points next to "=" or ":" code
30383 00001982 0000                   _$P_Save_EOB:         dw  0             ;AN000; save pointer to EOB
30384 00001984 0000                   _$P_Found_SYNONYM: dw     0             ;AN000; es:@ points to found synonym
30385
30386 00001986 00<rep 80h>            _$P_STRING_BUF:       times 128 db 0    ;AN000; Pick a operand from command line
30387                                 _$P_STRING_BUF_END equ    $             ;AN000;
30388
30389                                 ; 25/10/2022
30390                                 ; (MSDOS 5.0 IO.SYS, SYSINIT:16F8h)
30391
30392 00001A06 FF                     _$P_Char_CAP_Ptr:  db     0FFh          ;AN000; info id
30393 00001A07 0000                                dw  0             ;AN000; offset of char case map table
30394 00001A09 0000                                dw  0             ;AN000; segment of char case map table
30395                                 ; 25/10/2022
30396                                 ;IF CAPSW
30397                                 ;_$P_File_CAP_Ptr: db     0FFh          ;AN000; info id
30398                                 ;             dw  0             ;AN000; offset of file case map table
30399                                 ;             dw  0             ;AN000; segment of file case map table
30400                                 ;ENDIF
30401
30402                                 ; (tm06) IF FileSW                ;AN000;(Check if file spec is supported)
30403                                 ;
30404
30405                                 ;M029
30406                                 ;!!!WARNING!!!
30407                                 ; In routine SYSPARSE (parse.asm), _$P_FileSp_Char is reinitialized using
30408                                 ;hardcoded strings. If the chars in the string are changed here, corresponding
30409                                 ;changes need to be made in SYSPARSE
30410
30411                                 ;IF FileSW+DrvSW                  ;AN000;(Check if file spec is supported)
30412
30413                                 ; 25/10/2022
30414                                 ; (MSDOS 5.0 IO.SYS, SYSINIT:16FDh)
30415
30416 00001A0B 5B5D7C3C3E2B3D3B22       _$P_FileSp_Char       db  '[]|<>+=;"'     ;AN000; delimitter of file spec
30417                                 _$P_FileSp_Len        equ $-_$P_FileSp_Char ;AN000;
30418
30419                                 ;ENDIF                                  ;AN000;(of FileSW)
30420
30421                                 ; delimiter parsing
30422                                 _$P_colon_period   equ    01h           ;AN032; check for colon & period
30423                                 _$P_period_only       equ 02h           ;AN032; check only for period
30424
30425                                 ;filespec error flag
30426 00001A14 00                     _$P_err_flag:         db  0             ;AN033; flag set if filespec parsing error
30427                                                               ;AN033;  was detected.
30428                                 _$P_error_filespec equ    01h           ;AN033; mask to set flag
30429
30430
30431                                 ; PARSE.ASM - MSDOS 6.0 - 1991
30432                                 ; ============================================================================
30433                                 ; 27/03/2019 - Retro DOS v4.0
30434
30435                                 ;****************************************************************************
30436                                 ; SysParse;
30437                                 ;
30438                                 ;  Function : Parser Entry
30439                                 ;
30440                                 ;  Input: DS:SI -> command line
30441                                 ;      ES:DI -> parameter block
30442                                 ;      cs -> psdata.inc
30443                                 ;      CX = operand ordinal
30444                                 ;
30445                                 ;      Note:  ES is the segment containing all the control blocks defined
30446                                 ;             by the caller, except for the DOS COMMAND line parms, which
30447                                 ;             is in DS.
30448                                 ;
30449                                 ;  Output: CY = 1   error of caller, means invalid parameter block or
30450                                 ;                   invalid value list. But this parser does NOT implement
30451                                 ;                   this feature. Therefore CY always zero.
30452                                 ;
30453                                 ;       CY = 0   AX = return code
30454                                 ;                BL = terminated delimiter code
30455                                 ;                CX = new operand ordinal
30456                                 ;                SI = set past scaned operand
30457                                 ;                DX = selected result buffer
30458                                 ;
30459                                 ; Use:      _$P_Skip_Delim, _$P_Chk_EOL, _$P_Chk_Delim, _$P_Chk_DBCS
30460                                 ;   _$P_Chk_Swtch, _$P_Chk_Pos_Control, _$P_Chk_Key_Control
30461                                 ;   _$P_Chk_Sw_Control, _$P_Fill_Result
30462                                 ;
30463                                 ; Vars: _$P_Ordinal(RW), _$P_RC(RW), _$P_SI_Save(RW), _$P_DX(R), _$P_Terminator(R)
30464                                 ;   _$P_SaveSI_Cmpx(W), _$P_Flags(RW), _$P_Found_SYNONYM(R), _$P_Save_EOB(W)
30465                                 ;
30466                                 ;-------- Modification History ----------------------------------------
30467                                 ;
30468                                 ;  4/04/87 : Created by K. K,
30469                                 ;  4/28/87 : _$P_Val_YH assemble error (tm01)
30470                                 ;       : JMP SHORT assemble error (tm02)
30471                                 ;  5/14/87 : Someone doesn't want to include psdata (tm03)
30472                                 ;  6/12/87 : _$P_Bridge is missing when TimeSw equ 0 and (CmpxSw equ 1 or
30473                                 ;         DateSW equ 1)          (tm04)
30474                                 ;  6/12/87 : _$P_SorD_Quote is missing when QusSw equ 0 and CmpxSW equ 1
30475                                 ;                                (tm05) in PSDATA.INC
30476                                 ;  6/12/87 : _$P_FileSp_Char and _$P_FileSP_Len are missing
30477                                 ;         when FileSW equ 0 and DrvSW equ 1 (tm06) in PSDATA.INC
30478                                 ;  6/18/87 : $VAL1 and $VAL3, $VAL2 and $VAL3 can be used in the same
30479                                 ;         value-list block       (tm07)
30480                                 ;  6/20/87 : Add _$P_SW to check if there's an omiting parameter after
```

```
30481                           ;       switch (keyword) or not. If there is, backup si for next call
30482                           ;       (tm08)
30483                           ;  6/24/87 : Complex Item checking does not work correctly when CmpSW equ 1
30484                           ;       and DateSW equ 0 and TimeSW equ 0 (tm09)
30485                           ;  6/24/87 : New function flag _$P_colon_is_not_necessary for switch
30486                           ;       /+15 and /+:15 are allowed for user (tm10)
30487                           ;  6/29/87 : ECS call changes DS register but it causes the address problem
30488                           ;       in user's routines. _$P_Chk_DBCS (tm11)
30489                           ;  7/10/87 : Switch with no_match flag (0x0000H) does not work correctly
30490                           ;                                       (tm12)
30491                           ;  7/10/87 : Invalid switch/keyword does not work correctly
30492                           ;                                       (tm13)
30493                           ;  7/10/87 : Drive_only breaks 3 bytes after the result buffer
30494                           ;                                       (tm14)
30495                           ;  7/12/87 : Too_Many_Operands sets DX=0 as the PARSE result
30496                           ;                                       (tm15)
30497                           ;  7/24/87 : Negative lower bound on numeric ranges cause trouble
30498                           
30499                           ;  7/24/87 : Quoted strings being returned with quotes.
30500                           
30501                           ;  7/28/87 : Kerry S (;AN018;)
30502                           ;       Non optional value on switch (match flags<>0 and <>1) not flagged
30503                           ;       as an error when missing.       Solution: return error 2.  Modules
30504                           ;       affected: _$P_Chk_SW_Control.
30505                           
30506                           ;  7/29/87 : Kerry S (;AN019;)
30507                           ;       Now allow the optional bit in match flags for switches.  This
30508                           ;       allows the switch to be encountered with a value or without a
30509                           ;       value and no error is returned.
30510                           ;
30511                           
30512                           ;  8/28/87 : Ed K, Kerry S (;AN020;)
30513                           ;  9/14/87   In PROC _$P_Get_DecNum, when checking for field separators
30514                           ;       within a date response, instead of checking just for the one
30515                           ;       character defined by the COUNTRY DEPENDENT INFO, check for
30516                           ;       all three chars, "-", "/", and ".". Change _$P_Chk_Switch to allow
30517                           ;       slashes in date strings when DateSw (assembler switch) is set.
30518                           
30519                           ;  9/1/87  : Kerry S (;AN021)
30520                           ;       In PROC _$P_String_Comp, when comparing the switch or keyword on
30521                           ;       the command line with the string in the control block the
30522                           ;       comparing was stopping at a colon (switch) or equal (keyword)
30523                           ;       on the command line and assuming a match.      This allowed a shorter
30524                           ;       string on the command line than in the synonym list in the control
30525                           ;       block.  I put in a test for a null in the control block so the
30526                           ;       string in the control block must be the same length as the string
30527                           ;       preceeding the colon or equal on the command line.
30528                           
30529                           ;  8/28/87 : Kerry S (;AN022;)
30530                           ;       All references to data in PSDATA.INC had CS overrides.  This caused
30531                           ;       problems for people who included it themselves in a segment other
30532                           ;       than CS.  Added switch to allow including PSDATA.INC in any
30533                           ;       segment.
30534                           
30535                           ;  9/16/87 : Ed K (;AN023;) PTM1040
30536                           ;       in _$P_set_cdi PROC, it assumes CS points to psdata. Change Push CS
30537                           ;       into PUSH cs.  In _$P_Get_DecNum PROC, fix AN020
30538                           ;       forced both TIME and DATE to use the delims, "-","/",".".
30539                           ;       Created FLag, in _$P_time_Format PROC, to request the delim in
30540                           ;       BL be used if TIME is being parsed.
30541                           
30542                           ;  9/24/87 : Ed K
30543                           ;       Removed the include to STRUC.INC.      Replaced the STRUC macro
30544                           ;       invocations with their normally expanded code; made comments
30545                           ;       out of the STRUC macro invocation statements to maintain readability.
30546                           
30547                           ;  9/24/87 : Ed K (;AN024;) PTM1222
30548                           ;       When no CONTROL for a keyword found, tried to fill in RESULT
30549                           ;       pointed to by non-existant CONTROL.
30550                           
30551                           ; 10/15/87 : Ed K (;AN025;) PTM1672
30552                           ;       A quoted text string can be framed only by double quote.  Remove
30553                           ;       support to frame quoted text string with single quote.
30554                           ;       (apostrophe) _$P_SorD_Quote is removed from PSDATA.INC.
30555                           ;       _$P_SQuote EQU also removed from PSDATA.INC.  Any references to
30556                           ;       single quote in PROC prologues are left as is for history reasons.
30557                           
30558                           ;       This fixes another bug, not mentioned in p1672, in that two
30559                           ;       quote chars within a quoted string is supposed to be reported as
30560                           ;       one quote character, but is reported as two quotes.  This changed
30561                           ;       two instructions in PROC _$P_Quoted_Str.
30562                           
30563                           ;       Also fixed are several JMP that caused a NOP, these changed to
30564                           ;       have the SHORT operator to avoid the unneeded NOP.
30565                           
30566                           ;       The code and PSDATA.INC have been aligned for ease of reading.
30567                           
30568                           ; 10/26/87 : Ed K (;AN026;) PTM2041, DATE within SWITCH, BX reference to
30569                           ;       psdata buffer should have cs.
30570                           
30571                           ; 10/27/87 : Ed K (;AN027;) PTM2042 comma between keywords implies
30572                           ;       positional missing.
30573                           
30574                           ; 11/06/87 : Ed K (;AN028;) PTM 2315 Parser should not use line feed
30575                           ;       as a line delimiter, should use carriage return.
30576                           ;       Define switch: LFEOLSW, if on, accept LF as end of line char.
30577                           
30578                           ; 11/11/87 : Ed K (;AN029;) PTM 1651 GET RID OF WHITESPACE AROUND "=".
30579                           
30580                           ; 11/18/87 : Ed K (;AN030;) PTM 2551 If filename is just "", then
30581                           ;       endless loop since SI is returned still pointing to start
30582                           ;       of that parm.
30583                           
30584                           ; 11/19/87 : Ed K (;AN031;) PTM 2585 date & time getting bad values.
30585                           ;       Vector to returned string has CS instead of cs, but
30586                           ;       when tried to fix it on previous version, changed similar
30587                           ;       but wrong place.
30588                           
30589                           ; 12/09/87 : Bill L (;AN032;) PTM 2772 colon and period are now valid
30590                           ;       delimiters between hours, minutes, seconds for time. And period
30591                           ;       and comma are valid delimiters between seconds and 100th second.
30592                           
30593                           ; 12/14/87 : Bill L (;AN033;) PTM 2722 if illegal delimiter characters
30594                           ;       in a filespec, then flag an error.
30595                           
30596                           ; 12/22/87 : Bill L (;AN034;)         All local data to parser is now
30597                           ;       indexed off of the cs equate instead of the DS register.
30598                           ;       Using this method, DS can point to the segment of PSP or to psdata
30599                           ;  -->          local parser data. Why were some references to local data changed
30600                           ;       to do this before, but not all ?????
30601                           
30602                           ; 02/02/88 : Ed K (;AC035;) INSPECT utility, suggests optimizations.
30603                           
30604                           ; 02/05/88 : Ed K (;AN036;) P3372-UPPERCASE TRANSLATION, cs HOSED.
```

```
                                ;
                                ; 02/08/88 : Ed K (;AN037;) P3410-AVOID POP OF CS, CHECK BASESW FIRST.
                                
                                ; 02/19/88 : Ed K (;AN038;) p3524 above noon and "am" should be error
                                
                                ; 02/23/88 : Ed K (;AN039;) p3518 accept "comma" and "period" as decimal
                                ;        separator in TIME before hundredths field.
                                ;
                                ; 08/09/90 : SA    M005    Prevented parser from recognizing '=' signs within
                                ;                         strings as keywords.
                                ;
                                ;****************************************************************************
                                
                                ;IF FarSW                        ;AN000;(Check if need far return)
                                ;SysParse proc far               ;AN000;
                                ;ELSE                                   ;AN000;
                                ;SysParse proc near                     ;AN000;
                                ;ENDIF                                  ;AN000;(of FarSW)
                                
                                ; 27/03/2019 - Retro DOS v4.0
                                ; (MSDOS 6.21 IO.SYS - SYSINIT:1842h)
                                
                                ; 25/10/2022 - Retro DOS v4.0
                                ; (MSDOS 5.0 IO.SYS - SYSINIT:1707h)
                                
                                ; 06/09/2023 - Retro DOS v4.2 IO.SYS Optimization (& Retro DOS v5.0 pre-work)
                                ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:1D08h)
                                
                                SysParse:
                                    ; 06/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
                                    ; dx = 0
00001A15 1E                         push    ds ; *!*
00001A16 0E                         push    cs
00001A17 1F                         pop     ds
                                
                                    ;mov    word [cs:_$P_Flags],0 ;AC034; Clear all internal flags
                                    ;cld                          ;AN000; confirm forward direction
                                    ;mov    word [cs:_$P_ORDINAL],cx ;AC034; save operand ordinal
                                    ;mov    word [cs:_$P_RC],_$P_No_Error ;AC034; Assume no error
                                    ;mov    word [cs:_$P_Found_SYNONYM],0 ;AC034; initalize synonym pointer
                                    ;
                                    ;mov    word [cs:_$P_DX],0    ;AC034; (tm15)
                                
                                    ; 06/09/2023
00001A18 8916[7C19]                 mov     [_$P_Flags],dx ; 0    ;AC034; Clear all internal flags
00001A1C FC                         cld                           ;AN000; confirm forward direction
00001A1D 890E[6F19]                 mov     [_$P_ORDINAL],cx      ;AC034; save operand ordinal
00001A21 8916[7119]                 mov     [_$P_RC],dx ; $P_No_Error ;AC034; Assume no error
00001A25 8916[8419]                 mov     [_$P_Found_SYNONYM],dx; 0 ;AC034; initalize synonym pointer
00001A29 8916[7519]                 mov     [_$P_DX],dx ; 0               ;AC034; (tm15)
                                
                                ;M029 -- Begin changes
                                ; The table of special chars _$P_FileSp_Char should be initialized on every
                                ;entry to SysParse. This is in the non-checksum region and any program that
                                ;corrupts this table but does not corrupt the checksum region will leave
                                ;command.com parsing in an inconsistent state.
                                ; NB: The special characters string has been hardcoded here. If any change
                                ;is made to it in psdata.inc, a corresponding change needs to be made here.
                                
                                ;IF FileSW + DrvSW
                                    ; 14/04/2024 (NASM syntax BugFix) .. '][' (MASM) -> '[]' (NASM)
                                
                                    ;mov    word [cs:_$P_FileSp_Char], '[]'
                                    ;mov    word [cs:_$P_FileSp_Char+2], '|<'
                                    ;mov    word [cs:_$P_FileSp_Char+4], '>+'
                                    ;mov    word [cs:_$P_FileSp_Char+6], '=;'
                                
                                    ; 14/04/2024
                                    ; 06/09/2023
00001A2D C706[0B1A]5B5D             mov     word [_$P_FileSp_Char], '[]'  ; mov word [_$P_FileSp_Char],5D5Bh
00001A33 C706[0D1A]7C3C             mov     word [_$P_FileSp_Char+2], '|<'       ; 3C7Ch
00001A39 C706[0F1A]3E2B             mov     word [_$P_FileSp_Char+4], '>+'       ; 2B3Eh
00001A3F C706[111A]3D3B             mov     word [_$P_FileSp_Char+6], '=;'       ; 3B3Dh
                                ;ENDIF
                                    ; 06/09/2023
00001A45 1F                         pop     ds ; *!*
                                
                                ;M029 -- End of changes
                                
00001A46 E87806                     call    _$P_Skip_Delim        ;AN000; Move si to 1st non white space
00001A49 7313                       jnc     short _$P_Start               ;AN000; If EOL is not encountered, do parse
                                ;---------------------------- End of Line
00001A4B B8FFFF                     mov     ax,_$P_RC_EOL         ;AN000; set exit code to -1
00001A4E 53                         push    bx                    ;AN000;
                                    ;mov    bx,[es:di+_$P_PARMS_Blk.PARMSX_Address]
                                                                  ;AN000; Get the PARMSX address to
00001A4F 268B1D                     mov     bx,[es:di]
                                    ;cmp    cl,[es:bx+_$P_PARMSX_Blk.MinP]
                                                                  ;AN000; check ORDINAL to see if the minimum
00001A52 263A0F                     cmp     cl,[es:bx]
00001A55 7303                       jae     short _$P_Fin         ;AN000; positional found.
                                
00001A57 B80200                     mov     ax,_$P_Op_Missing     ;AN000; If no, set exit code to missing operand
                                _$P_Fin:                          ;AN000;
00001A5A 5B                         pop     bx                    ;AN000;
00001A5B E90A01                     jmp     _$P_Single_Exit               ;AN000; return to the caller
                                ;---------------------------
                                _$P_Start:                        ;AN000;
00001A5E 2E8936[7E19]               mov     [cs:_$P_SaveSI_Cmpx],si ;AN000;AC034; save ptr to command line for later use by complex,
00001A63 53                         push    bx                    ;AN000; quoted string or file spec.
00001A64 57                         push    di                    ;AN000;
00001A65 55                         push    bp                    ;AN000;
                                    ;;lea   bx,[cs:_$P_STRING_BUF] ;AC034; set buffer to copy from command string
                                    ; 02/11/2022
                                    ;lea    bx,[_$P_STRING_BUF]
                                    ; 07/09/2023
00001A66 BB[8619]                   mov     bx,_$P_STRING_BUF
00001A69 2EF606[7D19]20             test    byte [cs:_$P_Flags2],_$P_Extra ;AC034; 3/9 extra delimiter encountered ?
00001A6F 7543                       jnz     short _$P_Pack_End    ;AN000; 3/9 if yes, no need to copy
                                
                                _$P_Pack_Loop:                            ;AN000;
00001A71 AC                         lodsb                         ;AN000; Pick a operand from buffer
00001A72 E8F106                     call    _$P_Chk_Switch        ;AN000; Check switch character
00001A75 723C                       jc      short _$P_Pack_End_BY_EOL ;AN020; if carry set found delimiter type slash, need backup si, else
continue
                                
00001A77 E86906                     call    _$P_Chk_EOL           ;AN000; Check EOL character
00001A7A 7437                       je      short _$P_Pack_End_BY_EOL ;AN000; need backup si
                                
00001A7C E89906                     call    _$P_Chk_Delim         ;AN000; Check delimiter
00001A7F 7518                       jne     short _$P_PL01                ;AN000; If no, process next byte
                                
00001A81 2EF606[7D19]20             test    byte [cs:_$P_Flags2],_$P_Extra ;AC034; 3/9 If yes and white spec,
                                ; (tm08)jne short _$P_Pack_End    ;AN000; 3/9 then
```

```
30728 00001A87 7505                     jnz     short _$P_Pack_End_backup_si ;AN000; (tm08)
30729
30730 00001A89 E83506                   call    _$P_Skip_Delim          ;AN000; skip subsequent white space,too
30731 00001A8C EB26                     jmp     short _$P_Pack_End      ;AN000; finish copy by placing NUL at end
30732
30733                                 _$P_Pack_End_backup_si:                 ;AN000; (tm08)
30734 00001A8E 2EF606[7D19]41           test    byte [cs:_$P_Flags2],_$P_SW+_$P_equ ;AN000;AC034;  (tm08)
30735 00001A94 741E                     jz      short _$P_Pack_End      ;AN000; (tm08)
30736
30737 00001A96 4E                       dec     si                      ;AN000; (tm08)
30738 00001A97 EB1B                     jmp     short _$P_Pack_End      ;AN025; (tm08)
30739
30740                                 _$P_PL01:                           ;AN000;
30741 00001A99 2E8807                   mov     [cs:bx],al              ;AN000; move byte to STRING_BUF
30742 00001A9C 3C3D                     cmp     al,_$P_Keyword  ;'='    ;AN000; if it is equal character,
30743 00001A9E 7506                     jne     short _$P_PL00                  ;AN000; then
30744
30745 00001AA0 2E800E[7D19]01           or      byte [cs:_$P_Flags2],_$P_equ ;AC034; remember it in flag
30746                                 _$P_PL00:                           ;AN000;
30747 00001AA6 43                       inc     bx                      ;AN000; ready to see next byte
30748 00001AA7 E8D506                   call    _$P_Chk_DBCS            ;AN000; was it 1st byte of DBCS ?
30749 00001AAA 73C5                     jnc     short _$P_Pack_Loop     ;AN000; if no, process to next byte
30750
30751 00001AAC AC                       lodsb                           ;AN000; if yes, store
30752 00001AAD 2E8807                   mov     [cs:bx],al              ;AN000;    2nd byte of DBCS
30753 00001AB0 43                       inc     bx                      ;AN000; update pointer
30754 00001AB1 EBBE                     jmp     short _$P_Pack_Loop     ;AN000; process to next byte
30755
30756                                 _$P_Pack_End_BY_EOL:                    ;AN000;
30757 00001AB3 4E                       dec     si                      ;AN000; backup si pointer
30758                                 _$P_Pack_End:                           ;AN000;
30759 00001AB4 2E8936[7319]             mov     [cs:_$P_SI_Save],si     ;AC034; save next pointer, SI
30760                                   ; 07/09/2023
30761                                   ;mov    byte [cs:bx],_$P_NULL ;AN000; put NULL at the end
30762 00001AB9 30E4                     xor     ah,ah ; 0 ; *
30763 00001ABB 2E8827                   mov     [cs:bx],ah ; _$P_NULL ;AN000; put NULL at the end
30764                                   ;
30765 00001ABE 2E891E[8219]             mov     [cs:_$P_Save_EOB],bx    ;AC034; 3/17/87 keep the address for later use of complex
30766                                   ;mov    bx,[es:di+_$P_PARMS_Blk.PARMSX_Address] ;AN000; get PARMSX address
30767 00001AC3 268B1D                   mov     bx,[es:di]
30768                                   ;;lea   si,[cs:_$P_STRING_BUF];AC034;
30769                                   ; 02/11/2022
30770                                   ;lea    si,[_$P_STRING_BUF]
30771                                   ; 07/09/2023
30772 00001AC6 BE[8619]                 mov     si,_$P_STRING_BUF
30773 00001AC9 2E803C2F                 cmp     byte [cs:si],_$P_Switch ;AN000; the operand begins w/ switch char ?
30774 00001ACD 7440                     je      short _$P_SW_Manager  ;AN000; if yes, process as switch
30775
30776 00001ACF 2E803C22                 cmp     byte [cs:si],_$P_DQuote       ;M005;is it a string?
30777 00001AD3 7408                     je      short _$P_Positional_Manager ;M005;if so, process as one!
30778
30779 00001AD5 2EF606[7D19]01           test    byte [cs:_$P_Flags2],_$P_equ ;AC034; the operand includes equal char ?
30780 00001ADB 7552                     jnz     short _$P_Key_Manager ;AN000; if yes, process as keyword
30781
30782                                 _$P_Positional_Manager:                 ;AN000; else process as positional
30783 00001ADD 268A4701                 mov     al,[es:bx+_$P_PARMSX_Blk.MaxP] ;AN000; get maxp
30784                                   ; 07/09/2023
30785                                   ;xor    ah,ah                   ;AN000; ax = maxp
30786 00001AE1 2E3906[6F19]             cmp     [cs:_$P_ORDINAL],ax     ;AC034; too many positional ?
30787 00001AE6 7312                     jae     short _$P_Too_Many_Error ;AN000; if yes, set exit code to too many
30788
30789 00001AE8 2EA1[6F19]               mov     ax,[cs:_$P_ORDINAL]     ;AC034; see what the current ordinal
30790 00001AEC D1E0                     shl     ax,1                    ;AN000; ax = ax*2
30791 00001AEE 43                       inc     bx                      ;AC035; add '2' to
30792 00001AEF 43                       inc     bx                      ;AC035;  BX reg
30793                                                                   ;AN000; now bx points to 1st CONTROL
30794 00001AF0 01C3                     add     bx,ax                   ;AN000; now bx points to specified CONTROL address
30795 00001AF2 268B1F                   mov     bx,[es:bx]              ;AN000; now bx points to specified CONTROL itself
30796 00001AF5 E87200                   call    _$P_Chk_Pos_Control     ;AN000; Do process for positional
30797 00001AF8 EB53                     jmp     short _$P_Return_to_Caller ;AN000; and return to the caller
30798
30799                                 _$P_Too_Many_Error:                     ;AN000;
30800 00001AFA 2EC706[7119]0100         mov     word [cs:_$P_RC],_$P_Too_Many ;AC034; set exit code
30801 00001B01 EB4A                     jmp     short _$P_Return_to_Caller ;AN000; and return to the caller
30802
30803                                   ; 07/09/2023 - Retro DOSD v4.2 IO.SYS (Optimization)
30804                                   ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:1E06h)
30805                                 get_maxp:
30806                                   ;mov    al,[es:bx+1]
30807 00001B03 268A4701                 mov     al,[es:bx+_$P_PARMSX_Blk.MaxP] ;AN000; get maxp
30808                                   ; 07/09/2023
30809                                   ; ah=0 ; *
30810                                   ;xor    ah,ah ; 0               ;AN000; ax = maxp
30811 00001B07 30ED                     xor     ch,ch ; **
30812 00001B09 40                       inc     ax                      ;AN000;
30813 00001B0A D1E0                     shl     ax,1                    ;AN000; ax = (ax+1)*2
30814 00001B0C 01C3                     add     bx,ax                   ;AN000; now bx points to maxs
30815 00001B0E C3                       retn
30816
30817                                 _$P_SW_Manager:                         ;AN000;
30818                                   ; 07/09/2023
30819                                   ;mov    al,[es:bx+_$P_PARMSX_Blk.MaxP] ;AN000; get maxp
30820                                   ;xor    ah,ah                   ;AN000; ax = maxp
30821                                   ;inc    ax                      ;AN000;
30822                                   ;shl    ax,1                    ;AN000; ax = (ax+1)*2
30823                                   ;add    bx,ax                   ;AN000; now bx points to maxs
30824 00001B0F E8F1FF                   call    get_maxp ; 07/09/2023
30825
30826 00001B12 268A0F                   mov     cl,[es:bx]              ;AN000;
30827                                   ; 07/09/2023
30828                                   ;xor    ch,ch ; ** (ch=0)       ;AN000; cx = maxs
30829                                   ;or     cx,cx                   ;AN000; at least one switch ?
30830                                   ;jz     short _$P_SW_Not_Found          ;AN000;
30831                                   ; 07/07/2023
30832 00001B15 E30F                     jcxz    _$P_SW_Not_Found        ; no
30833
30834 00001B17 43                       inc     bx                      ;AN000; now bx points to 1st CONTROL address
30835
30836                                 _$P_SW_Mgr_Loop:                        ;AN000;
30837 00001B18 53                       push    bx                      ;AN000;
30838 00001B19 268B1F                   mov     bx,[es:bx]              ;AN000; bx points to Switch CONTROL itself
30839 00001B1C E8A900                   call    _$P_Chk_SW_Control      ;AN000; do process for switch
30840 00001B1F 5B                       pop     bx                      ;AN000;
30841 00001B20 732B                     jnc     short _$P_Return_to_Caller ;AN000; if the CONTROL is for the switch, exit
30842
30843 00001B22 43                       inc     bx                      ;AC035; add '2' to
30844 00001B23 43                       inc     bx                      ;AC035;  BX reg
30845                                                                   ;AN000; else bx points to the next CONTROL
30846 00001B24 E2F2                     loop    _$P_SW_Mgr_Loop                 ;AN000; and loop
30847
30848                                 _$P_SW_Not_Found:                       ;AN000;
30849 00001B26 2EC706[7119]0300         mov     word [cs:_$P_RC],_$P_Not_In_SW ;AC034; here no CONTROL for the switch has
30850 00001B2D EB1E                     jmp     short _$P_Return_to_Caller ;AN000; not been found, means error.
30851
```

```
30852                                   _$P_Key_Manager:                ;AN000;
30853                                       ; 07/09/2023
30854                                       ;mov    al,[es:bx+_$P_PARMSX_Blk.MaxP] ;AN000; get maxp
30855                                       ;xor    ah,ah                   ;AN000; ax = maxp
30856                                       ;inc    ax                      ;AN000;
30857                                       ;shl    ax,1                    ;AN000; ax = (ax+1)*2
30858                                       ;add    bx,ax                   ;AN000; now bx points to maxs
30859 00001B2F E8D1FF                     call    get_maxp ; 07/09/2023
30860
30861 00001B32 268A07                      mov     al,[es:bx]              ;AN000;
30862 00001B35 30E4                        xor     ah,ah ; 0               ;AN000; ax = maxs
30863 00001B37 D1E0                        shl     ax,1                    ;AN000;
30864 00001B39 40                          inc     ax                      ;AN000; ax = ax*2+1
30865 00001B3A 01C3                        add     bx,ax                   ;AN000; now bx points to maxk
30866 00001B3C 268A0F                      mov     cl,[es:bx]              ;AN000;
30867                                      ; 07/09/2023
30868                                      ;xor    ch,ch ; ** (ch=0)       ;AN000; cx = maxk
30869                                      ;or     cx,cx                   ;AN000; at least one keyword ?
30870                                      ;jz     short _$P_Key_Not_Found       ;AN000;
30871                                      ; 07/07/2023
30872 00001B3F E305                        jcxz    _$P_Key_Not_Found       ; no
30873
30874 00001B41 43                          inc     bx                      ;AN000; now bx points to 1st CONTROL
30875
30876                                   _$P_Key_Mgr_Loop:               ;AN000;
30877                                       ; 07/09/2023
30878                                       ; ('_$P_Chk_Key_Control' contains only 'stc' instruction)
30879                                       ; (always returns with cf=1)
30880                                       ;push   bx                      ;AN000;
30881                                       ;mov    bx,[es:bx]              ;AN000; bx points to keyword CONTROL itself
30882                                       ;call   _$P_Chk_Key_Control     ;AN000; do process for keyword
30883                                       ;pop    bx                      ;AN000;
30884                                       ;jnc    short _$P_Return_to_Caller ;AN000; if the CONTROL is for the keyword, exit
30885                                       ; 07/09/2023
30886                                       ; cf=1 (after 'call _$P_Chk_Key_Control')
30887
30888 00001B42 43                          inc     bx                      ;AC035; add '2' to
30889 00001B43 43                          inc     bx                      ;AC035;  BX reg
30890                                                                      ;AN000; else bx points to the next CONTROL
30891 00001B44 E2FC                        loop    _$P_Key_Mgr_Loop        ;AN000; and loop
30892
30893                                   _$P_Key_Not_Found:              ;AN000;
30894 00001B46 2EC706[7119]0400            mov     word [cs:_$P_RC],_$P_Not_In_Key ;AC034; here no CONTROL for the keyword has
30895                                   _$P_Return_to_Caller:                   ;AN000;
30896 00001B4D 5D                          pop     bp                      ;AN000;
30897 00001B4E 5F                          pop     di                      ;AN000;
30898 00001B4F 5B                          pop     bx                      ;AN000;
30899 00001B50 2E8B0E[6F19]                mov     cx,[cs:_$P_ORDINAL]     ;AC034; return next ordinal
30900 00001B55 2EA1[7119]                  mov     ax,[cs:_$P_RC]          ;AC034; return exit code
30901 00001B59 2E8B36[7319]                mov     si,[cs:_$P_SI_Save]     ;AC034; return next operand pointer
30902 00001B5E 2E8B16[7519]                mov     dx,[cs:_$P_DX]          ;AC034; return result buffer address
30903 00001B63 2E8A1E[7719]                mov     bl,[cs:_$P_Terminator];AC034; return delimiter code found
30904                                   _$P_Single_Exit:                ;AN000;
30905 00001B68 F8                          clc                             ;AN000;
30906 00001B69 C3                          retn                            ;AN000;
30907
30908                                   ;****************************************************************************
30909                                   ; _$P_Chk_Pos_Control
30910                                   ;
30911                                   ; Function: Parse CONTROL block for a positional
30912                                   ;
30913                                   ; Input:     ES:BX -> CONTROL block
30914                                   ;        cs:SI -> _$P_STRING_BUF
30915                                   ;
30916                                   ; Output:    None
30917                                   ;
30918                                   ; Use:       _$P_Fill_Result, _$P_Check_Match_Flags
30919                                   ;
30920                                   ; Vars: _$P_Ordinal(W), _$P_RC(W)
30921                                   ;****************************************************************************
30922
30923                                   _$P_Chk_Pos_Control:
30924 00001B6A 50                          push    ax                      ;AN000;
30925                                       ;mov    ax,[es:bx+_$P_Control_Blk.Match_Flag] ;AN000;
30926 00001B6B 268B07                      mov     ax,[es:bx]
30927                                      ; 12/12/2022
30928 00001B6E A802                        test    al,_$P_Repeat
30929                                      ;test   ax,_$P_Repeat           ;AN000; repeat allowed ?
30930 00001B70 7505                        jnz     short _$P_CPC00                 ;AN000; then do not increment ORDINAL
30931
30932 00001B72 2EFF06[6F19]                inc     word [cs:_$P_ORDINAL] ;AC034; update the ordinal
30933                                   _$P_CPC00:                      ;AN000;
30934 00001B77 2E803C00                    cmp     byte [cs:si],_$P_NULL ;AN000; no data ?
30935 00001B7B 7517                        jne     short _$P_CPC01                 ;AN000;
30936
30937                                      ; 12/12/2022
30938 00001B7D A801                        test    al,_$P_Optional
30939                                      ;test   ax,_$P_Optional                 ;AN000; yes, then is it optional ?
30940 00001B7F 7509                        jnz     short _$P_CPC02                 ;AN000;
30941
30942 00001B81 2EC706[7119]0200            mov     word [cs:_$P_RC],_$P_Op_Missing ;AC034; no, then error 3/17/87
30943 00001B88 EB0D                        jmp     short _$P_CPC_Exit      ;AN000;
30944
30945                                   _$P_CPC02:                      ;AN000;
30946 00001B8A 50                          push    ax                      ;AN000;
30947                                       ;mov    al,_$P_String           ;AN000; if it is optional return NULL
30948                                       ;mov    ah,_$P_No_Tag           ;AN000; no item tag indication
30949                                       ; 07/07/2023
30950 00001B8B B803FF                      mov     ax,(_$P_No_Tag<<8)|_$P_String
30951 00001B8E E89600                      call    _$P_Fill_Result                 ;AN000;
30952 00001B91 58                          pop     ax                      ;AN000;
30953 00001B92 EB03                        jmp     short _$P_CPC_Exit      ;AN000;
30954
30955                                   _$P_CPC01:                      ;AN000;
30956 00001B94 E81101                      call    _$P_Check_Match_Flags ;AN000;
30957                                   _$P_CPC_Exit:                           ;AN000;
30958 00001B97 58                          pop     ax                      ;AN000;
30959 00001B98 C3                          retn                            ;AN000;
30960
30961                                   ;****************************************************************************
30962                                   ; _$P_Chk_Key_Control
30963                                   ;
30964                                   ; Function: Parse CONTROL block for a keyword
30965                                   ;
30966                                   ; Input:     ES:BX -> CONTROL block
30967                                   ;        cs:SI -> _$P_STRING_BUF
30968                                   ;
30969                                   ; Output:    CY = 1 : not match
30970                                   ;
30971                                   ; Use:       _$P_Fill_Result, _$P_Search_KEYorSW, _$P_Check_Match_Flags
30972                                   ;
30973                                   ; Vars: _$P_RC(W), _$P_SaveSI_Cmpx(W), _$P_KEYorSW_Ptr(R), _$P_Flags(W)
30974                                   ;****************************************************************************
30975
```

```
                                ; 07/09/2023
                                ;_$P_Chk_Key_Control:
                                ;   stc                         ;AN000; this logic works when the KeySW
                                ;   retn                        ;AN000; is reset.
                                ;
                                ;//////////////////////////////////////////////////////////////////////
                                ;
                                ; _$P_Search_KEYorSW:
                                ;
                                ; Function: Seach specified keyword or switch from CONTROL
                                ;
                                ; Input:     ES:BX -> CONTROL block
                                ;        cs:SI -> _$P_STRING_BUF
                                ;
                                ; Output:    CY = 1 : not match
                                ;
                                ; Use:       _$P_String_Comp, _$P_MoveBP_NUL, _$P_Found_SYNONYM
                                ;//////////////////////////////////////////////////////////////////////
                                ;
                                    ; 25/10/2022 - Retro DOS v4.0
                                    ; (MSDOS 5.0 IO.SYS - SYSINIT:18B6h)
                                ;
                                _$P_Search_KEYorSW:                     ;AN000;
00001B99 55                         push    bp                  ;AN000;
00001B9A 51                         push    cx                  ;AN000;
00001B9B 268A4F08                   mov     cl,[es:bx+_$P_Control_Blk.nid] ;AN000; Get synonym count
00001B9F 30ED                       xor     ch,ch               ;AN000; and set it to cx
                                    ;or     cx,cx               ;AN000; No synonyms specified ?
                                    ;jz     short _$P_KEYorSW_Not_Found ;AN000; then indicate not found by CY
                                    ; 07/07/2023
00001BA1 E30D                       jcxz    _$P_KEYorSW_Not_Found
                                    ;lea    bp,[es:bx+_$P_Control_Blk.KEYorSW] ;AN000; BP points to the 1st synonym
                                    ; 25/10/2022
00001BA3 8D6F09                     lea     bp,[bx+_$P_Control_Blk.KEYorSW]
                                    ;lea    bp,[bx+9]
                                _$P_KEYorSW_Loop:                   ;AN000;
00001BA6 E8B503                     call    _$P_String_Comp         ;AN000; compare string in buffer w/ the synonym
00001BA9 7308                       jnc     short _$P_KEYorSW_Found ;AN000; If match, set it to synonym pointer
                                
00001BAB E80E00                     call    _$P_MoveBP_NUL          ;AN000; else, bp points to the next string
00001BAE E2F6                       loop    _$P_KEYorSW_Loop        ;AN000; loop nid times
                                _$P_KEYorSW_Not_Found:                  ;AN000;
00001BB0 F9                         stc                             ;AN000; indicate not found in synonym list
00001BB1 EB06                       jmp     short _$P_KEYorSW_Exit;AN000; and exit
                                
                                _$P_KEYorSW_Found:                  ;AN000;
00001BB3 2E892E[8419]               mov     [cs:_$P_Found_SYNONYM],bp ;AC034; set synonym pointer
00001BB8 F8                         clc                             ;AN000; indicate found
                                _$P_KEYorSW_Exit:                   ;AN000;
00001BB9 59                         pop     cx                      ;AN000;
00001BBA 5D                         pop     bp                      ;AN000;
00001BBB C3                         retn                            ;AN000;
                                
                                ;//////////////////////////////////////////////////////////////////////
                                ; _$P_MoveBP_NUL
                                ;//////////////////////////////////////////////////////////////////////
                                
                                _$P_MoveBP_NUL:
                                _$P_MBP_Loop:                               ;AN000;
                                    ; 11/12/2022
00001BBC 26807E0000                 cmp     byte [es:bp],_$P_NULL ;AN000; Increment BP that points
                                    ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
                                    ; (SYSINIT:18DBh)
                                    ;cmp     byte [es:bp+0],0
00001BC1 7403                       je      short _$P_MBP_Exit      ;AN000; to the synomym list
                                
00001BC3 45                         inc     bp                      ;AN000; until
00001BC4 EBF6                       jmp     short _$P_MBP_Loop      ;AN000; NULL encountered.
                                
                                _$P_MBP_Exit:                               ;AN000;
00001BC6 45                         inc     bp                      ;AN000; bp points to next to NULL
00001BC7 C3                         retn                            ;AN000;
                                
                                ;//////////////////////////////////////////////////////////////////////
                                ; _$P_Chk_SW_Control
                                ;
                                ; Function: Parse CONTROL block for a switch
                                ;
                                ; Input:     ES:BX -> CONTROL block
                                ;        cs:SI -> _$P_STRING_BUF
                                ;
                                ; Output:    CY = 1 : not match
                                ;
                                ; Use:       _$P_Fill_Result, _$P_Search_KEYorSW, _$P_Check_Match_Flags
                                ;
                                ; Vars:  _$P_SaveSI_Cmpx(W), _$P_KEYorSW_Ptr(R), _$P_Flags(W)
                                ;//////////////////////////////////////////////////////////////////////
                                
                                _$P_Chk_SW_Control:
                                
                                ;IF SwSW                             ;AN000;(Check if switch is supported)
                                    ;or     byte [cs:_$P_Flags+1],10h
00001BC8 2E800E[7D19]10             or      byte [cs:_$P_Flags2],_$P_SW_Cmp ;AC034; Indicate switch for later string comparison
00001BCE E8C8FF                     call    _$P_Search_KEYorSW      ;AN000; Search the switch in the CONTROL block
00001BD1 7248                       jc      short _$P_Chk_SW_Err0 ;AN000; not found, then try next CONTROL
                                
                                    ;and    [cs:_$P_Flags+],0EFh
00001BD3 2E8026[7D19]EF             and     byte [cs:_$P_Flags2],0FFh-_$P_SW_Cmp
                                                                    ;AC034; reset the indicator previously set
00001BD9 50                         push    ax                      ;AN000;             /switch:
00001BDA 2EA1[8019]                 mov     ax,[cs:_$P_KEYorSW_Ptr] ;AC034;             ^       ^
00001BDE 29F0                       sub     ax,si                   ;AN000;  SI    KEYorSW
00001BE0 2E0106[7E19]               add     [cs:_$P_SaveSI_Cmpx],ax     ;AC034; update for complex list
00001BE5 58                         pop     ax                      ;AN000;
                                
00001BE6 2E8B36[8019]               mov     si,[cs:_$P_KEYorSW_Ptr] ;AC034; set si at the end or colon
00001BEB 2E803C00                   cmp     byte [cs:si],_$P_NULL ;AN000; any data after colon
00001BEF 7525                       jne     short _$P_CSW00             ;AN000; if yes, process match flags
                                
00001BF1 2E807CFF3A                 cmp     byte [cs:si-1],_$P_Colon ;AN000; if no, the switch terminated by colon ?
00001BF6 7509                       jne     short _$P_Chk_if_data_required ;AN000; if yes,
                                
00001BF8 2EC706[7119]0900           mov     word [cs:_$P_RC],_$P_Syntax ;AC034; return syntax error
00001BFF EB1C                       jmp     short _$P_Chk_SW_Exit ;AN000;
                                
                                _$P_Chk_if_data_required:           ;AN018; no data, no colon
                                    ;cmp    word [es:bx+_$P_Control_Blk.Match_Flag],0
00001C01 26833F00                   cmp     word [es:bx],0          ;AN018; should have data? zero match flag means switch followed by nothing is
OK
00001C05 7416                       je      short _$P_Chk_SW_Exit ;AN018; match flags not zero so should have something if optional bit is not
on
                                
                                    ;test   word [es:bx+_$P_Control_Blk.Match_Flag],_$P_Optional
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYINIT compatibility)
```

```
                                    ;test  word [es:bx],1
                                    ; 12/12/2022
                                    ;test  word [es:bx],_$P_Optional ;AN019; see if no value is valid
00001C07 26F60701                   test   byte [es:bx],_$P_Optional
00001C0B 7510                       jnz    short _$P_Chk_SW_Exit ;AN019; if so, then leave, else yell

00001C0D 2EC706[7119]0200           mov    word [cs:_$P_RC],_$P_Op_Missing ;AC034; return required operand missing
00001C14 EB07                       jmp    short _$P_Chk_SW_Exit ;AN018;

                                  _$P_CSW00:                        ;AN000;
00001C16 E88F00                     call   _$P_Check_Match_Flags ;AN000; process match flag
00001C19 F8                         clc                           ;AN000; indicate match
                                    ;jmp   short _$P_Chk_SW_Single_Exit ;AN000;
                                    ; 12/12/2022
00001C1A C3                         retn

                                  _$P_Chk_SW_Err0:                  ;AN000;
00001C1B F9                         stc                           ;AN000; not found in switch synonym list
                                    ;jmp   short _$P_Chk_SW_Single_Exit ;AN000;
                                    ; 12/12/2022
00001C1C C3                         retn

                                  _$P_Chk_SW_Exit:                  ;AN000;
00001C1D 50                         push   ax                     ;AN000;
                                    ;mov   al,_$P_String          ;AN000;
                                    ;mov   ah,_$P_No_Tag          ;AN000;
                                    ; 07/07/2023
00001C1E B803FF                     mov    ax,(_$P_No_Tag<<8)|_$P_String
00001C21 E80300                     call   _$P_Fill_Result               ;AN000; set result buffer
00001C24 58                         pop    ax                     ;AN000;
00001C25 F8                         clc                           ;AN000;
                                  _$P_Chk_SW_Single_Exit:                ;AN000;
00001C26 C3                         retn                          ;AN000;
                                  ;ELSE                                  ;AN000;(of IF SwSW)
                                  ;   stc                           ;AN000; this logic works when the SwSW
                                  ;   retn                          ;AN000; is reset.

                                  ;******************************************************************************
                                  ; _$P_Fill_Result
                                  ;
                                  ; Function: Fill the result buffer
                                  ;
                                  ; Input:    AH = Item tag
                                  ;       AL = type
                                  ;            AL = 1: CX,DX has 32bit number (CX = high)
                                  ;            AL = 2: DX has index(offset) into value list
                                  ;            AL = 6: DL has driver # (1-A, 2-B, ... , 26 - Z)
                                  ;            AL = 7: DX has year, CL has month and CH has date
                                  ;            AL = 8: DL has hours, DH has minutes, CL has seconds,
                                  ;                    amd CH has hundredths
                                  ;            AL = else: cs:SI points to returned string buffer
                                  ;       ES:BX -> CONTROL block
                                  ;
                                  ; Output:   None
                                  ;
                                  ; Use:     _$P_Do_CAPS_String, _$P_Remove_Colon, _$P_Found_SYNONYM
                                  ;
                                  ; Vars: _$P_DX(W)
                                  ;******************************************************************************

                                  _$P_Fill_Result:
00001C27 57                         push   di                     ;AN000;
00001C28 268B7F04                   mov    di,[es:bx+_$P_Control_Blk.Result_Buf]
                                                                  ;AN000; di points to result buffer
00001C2C 2E893E[7519]               mov    [cs:_$P_DX],di         ;AC034; set returned result address
                                    ;mov   [es:di+_$P_Result_Blk.Type],al ;AN000; store type
                                    ;mov   [es:di+_$P_Result_Blk.Item_Tag],ah ;AN000; store item tag
                                    ; 07/09/2023
                                    ;mov   [es:di+_$P_Result_Blk.Type], ax
00001C31 268905                     mov    [es:di],ax             ; store type (al) and item tag (ah)

00001C34 50                         push   ax                     ;AN000;
00001C35 2EA1[8419]                 mov    ax,[cs:_$P_Found_SYNONYM] ;AC034; if yes,
00001C39 26894502                   mov    [es:di+_$P_Result_Blk.SYNONYM_Ptr],ax
                                                                  ;AN000;   then set it to the result
00001C3D 58                         pop    ax                     ;AN000;
                                  _$P_RLT04:                      ;AN000;
00001C3E 3C01                       cmp    al,_$P_Number          ;AN000; if number
00001C40 750A                       jne    short _$P_RLT00               ;AN000;

                                  _$P_RLT02:                      ;AN000;
00001C42 26895504                   mov    [es:di+_$P_Result_Blk.Picked_Val],dx ;AN000; then store 32bit
00001C46 26894D06                   mov    [es:di+_$P_Result_Blk.Picked_Val+2],cx ;AN000; number
00001C4A EB5A                       jmp    short _$P_RLT_Exit     ;AN000;

                                  _$P_RLT00:                      ;AN000;
00001C4C 3C02                       cmp    al,_$P_List_Idx               ;AN000; if list index
00001C4E 7506                       jne    short _$P_RLT01               ;AN000;

00001C50 26895504                   mov    [es:di+_$P_Result_Blk.Picked_Val],dx
                                                                  ;AN000; then store list index
00001C54 EB50                       jmp    short _$P_RLT_Exit     ;AN000;

                                  _$P_RLT01:                      ;AN000;
00001C56 3C07                       cmp    al,_$P_Date_F          ;AN000; Date format ?
00001C58 74E8                       je     short _$P_RLT02               ;AN000;

00001C5A 3C08                       cmp    al,_$P_Time_F          ;AN000; Time format ?
00001C5C 74E4                       je     short _$P_RLT02               ;AN000;

00001C5E 3C06                       cmp    al,_$P_Drive           ;AN000; drive format ?
00001C60 7506                       jne    short _$P_RLT03               ;AN000;

00001C62 26885504                   mov    [es:di+_$P_Result_Blk.Picked_Val],dl ;AN000; store drive number
00001C66 EB3E                       jmp    short _$P_RLT_Exit     ;AN000;

                                  _$P_RLT03:                      ;AN000;
00001C68 3C04                       cmp    al,_$P_Complex         ;AN000; complex format ?
00001C6A 750F                       jne    short _$P_RLT05               ;AN000;

00001C6C 2EA1[7E19]                 mov    ax,[cs:_$P_SaveSI_Cmpx] ;AC034; then get pointer in command buffer
00001C70 40                         inc    ax                     ;AN000; skip left Parentheses
00001C71 26894504                   mov    [es:di+_$P_Result_Blk.Picked_Val],ax ;AN000; store offset
00001C75 268C5D06                   mov    [es:di+_$P_Result_Blk.Picked_Val+2],ds ;AN000; store segment
00001C79 EB2B                       jmp    short _$P_RLT_Exit     ;AN000;

                                  _$P_RLT05:                      ;AN000;
                                  ;-------------------------  AL = 3, 5, or 9
00001C7B 26897504                   mov    [es:di+_$P_Result_Blk.Picked_Val],si
                                                                  ;AN000; store offset of STRING_BUF
00001C7F 268C4D06                   mov    [es:di+_$P_Result_Blk.Picked_Val+2],cs
                                                                  ;AN031; store segment of STRING_BUF
00001C83 50                         push   ax                     ;AN000;
00001C84 26F6470201                 test   byte [es:bx+_$P_Control_Blk.Function_Flag],_$P_CAP_File
```

```
                                                        ;AN000; need CAPS by file table?
00001C89 7404                       jz      short _$P_RLT_CAP00   ;AN000;
                                    mov     al,_$P_DOSTBL_File    ;AN000; use file upper case table
00001C8B B004                       jmp     short _$P_RLT_CAP02   ;AN000;
00001C8D EB09
                                  _$P_RLT_CAP00:                          ;AN000;
00001C8F 26F6470202                 test    byte [es:bx+_$P_Control_Blk.Function_Flag],_$P_CAP_Char
                                                        ;AN000; need CAPS by char table ?
00001C94 7405                       jz      short _$P_RLT_CAP01   ;AN000;

00001C96 B002                       mov     al,_$P_DOSTBL_Char    ;AN000; use character upper case table
                                  _$P_RLT_CAP02:                          ;AN000;
00001C98 E8DF00                     call    _$P_Do_CAPS_String    ;AN000;  process CAPS along the table
                                  _$P_RLT_CAP01:                          ;AN000;
00001C9B 58                         pop     ax                    ;AN000;
00001C9C 26F6470210                 test    byte [es:bx+_$P_Control_Blk.Function_Flag],_$P_Rm_Colon
                                                        ;AN000; removing colon at end ?
00001CA1 7403                       jz      short _$P_RLT_Exit    ;AN000;

00001CA3 E8AE00                     call    _$P_Remove_Colon      ;AN000; then process it.
                                  _$P_RLT_Exit:                           ;AN000;
00001CA6 5F                         pop     di                    ;AN000;
00001CA7 C3                         retn                          ;AN000;

                                  ;****************************************************************************
                                  ; _$P_Check_Match_Flags
                                  ;
                                  ; Function:  Check the mutch_flags and make the exit code and set the
                                  ;        result buffer
                                  ;
                                  ;       Check for types in this order:
                                  ;          Complex
                                  ;          Date
                                  ;          Time
                                  ;          Drive
                                  ;          Filespec
                                  ;          Quoted String
                                  ;          Simple String
                                  ;
                                  ; Input:     cs:SI -> _$P_STRING_BUF
                                  ;         ES:BX -> CONTROL block
                                  ;
                                  ; Output:    None
                                  ;
                                  ; Use:           _$P_Value, P$_SValue, _$P_Simple_String, _$P_Date_Format
                                  ;         _$P_Time_Format, _$P_Complex_Format, _$P_File_Foemat
                                  ;         _$P_Drive_Format
                                  ;****************************************************************************

                                      ; 25/10/2022 - Retro DOS v4.0
                                      ; (MSDOS 5.0 IO.SYS - SYSINIT:19CFh)

                                      ; 14/04/2024 - Retro DOS v5.0
                                      ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:1FC3h)

                                      ; 12/12/2022
                                  _$P_Check_Match_Flags:
00001CA8 2EC606[141A]00             mov     byte [cs:_$P_err_flag],_$P_NULL
                                                        ;AN033;AC034;; clear filespec error flag.
00001CAE 50                         push    ax                    ;AN000;
                                    ;mov    ax,[es:bx+_$P_Control_Blk.Match_Flag]
00001CAF 268B07                     mov     ax,[es:bx]            ;AN000; load match flag(16bit) to ax
00001CB2 09C0                       or      ax,ax                 ;AC035; test ax for zero
00001CB4 7517                       jnz     short _$P_Mat         ;AN000; (tm12)
00001CB6 50                         push    ax                    ;AN000; (tm12)
00001CB7 53                         push    bx                    ;AN000; (tm12)
00001CB8 52                         push    dx                    ;AN000; (tm12)
00001CB9 57                         push    di                    ;AN000; (tm12)
00001CBA 2EC706[7119]0900           mov     word [cs:_$P_RC],_$P_Syntax ;AC034; (tm12)
                                    ;mov    ah,_$P_No_Tag         ;AN000; (tm12)
                                    ;mov    al,_$P_String         ;AN000; (tm12)
                                    ; 07/07/2023
00001CC1 B803FF                     mov     ax,(_$P_No_Tag<<8)|_$P_String
00001CC4 E860FF                     call    _$P_Fill_Result             ;AN000; (tm12)
00001CC7 5F                         pop     di                    ;AN000; (tm12)
00001CC8 5A                         pop     dx                    ;AN000; (tm12)
00001CC9 5B                         pop     bx                    ;AN000; (tm12)
00001CCA 58                         pop     ax                    ;AN000; (tm12)
                                    ; 12/12/2022
                                    ;jmp    short _$P_Bridge      ;AC035; (tm12)
                                    ; 12/12/2022
                                  ;_$P_Mat:                               ;AN000; (tm12)
                                    ;jmp    short _$P_Match03     ;AN025; (tm09)
                                  _$P_Bridge:
00001CCB EB6E                       jmp     short _$P_Match_Exit  ;AN000; (tm02)

                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ; (SYSINIT:19F9h)
                                    ; 12/12/2022
                                    ;nop    ; db 90h

                                  ; 12/12/2022
                                  _$P_Mat:
                                  _$P_Match03:                            ;AN000;
                                    ;test   ax,_$P_Num_Val ; 8000h;AN000; Numeric value
                                    ; 07/07/2023
00001CCD F6C480                     test    ah,(_$P_Num_Val>>8) ; 80h
00001CD0 7412                       jz      short _$P_Match04     ;AN000;

00001CD2 2EC706[7119]0000           mov     word [cs:_$P_RC],_$P_No_Error ;AC034; assume no error
00001CD9 E81E01                     call    _$P_Value             ;AN000; do process
00001CDC 2E833E[7119]09             cmp     word [cs:_$P_RC],_$P_Syntax ;AC034; if error, examine the next type
00001CE2 7557                       jne     short _$P_Match_Exit  ;AN000;
                                  _$P_Match04:                            ;AN000;
                                    ;test   ax,_$P_SNum_Val ; 4000h       ;AN000; Signed numeric value
                                    ; 07/07/2023
00001CE4 F6C440                     test    ah,(_$P_SNum_Val>>8) ; 40h
00001CE7 7412                       jz      short _$P_Match05     ;AN000;

00001CE9 2EC706[7119]0000           mov     word [cs:_$P_RC],_$P_No_Error ;AC034; assume no error
00001CF0 E8E300                     call    _$P_SValue            ;AN000; do process
00001CF3 2E833E[7119]09             cmp     word [cs:_$P_RC],_$P_Syntax ;AC034; if error, examine the next type
00001CF9 7540                       jne     short _$P_Match_Exit  ;AN000;
                                  _$P_Match05:                            ;AN000;
                                    ;test   ax,_$P_Drv_Only ; 100h;AN000; Drive only
                                    ; 07/07/2023
00001CFB F6C401                     test    ah,(_$P_Drv_Only>>8) ; 1
00001CFE 7415                       jz      short _$P_Match06     ;AN000;

00001D00 2EC706[7119]0000           mov     word [cs:_$P_RC],_$P_No_Error ;AC034; assume no error
00001D07 E8F202                     call    _$P_File_Format             ;AN000; 1st, call file format
00001D0A E87203                     call    _$P_Drive_Format      ;AN000; check drive format, next
00001D0D 2E833E[7119]09             cmp     word [cs:_$P_RC],_$P_Syntax ;AC034; if error, examine the next type
```

```
00001D13 7526                       jne     short _$P_Match_Exit  ;AN000;
                                  _$P_Match06:                              ;AN000;
                                      ;test   ax,_$P_File_Spc ; 200h;AN000; File spec
                                      ; 07/07/2023
00001D15 F6C402                       test    ah,(_$P_File_Spc>>8) ; 2
00001D18 7412                         jz      short _$P_Match07     ;AN000;

00001D1A 2EC706[7119]0000             mov     word [cs:_$P_RC],_$P_No_Error ;AC034; assume no error
00001D21 E8D802                       call    _$P_File_Format               ;AN000; do process
00001D24 2E833E[7119]09               cmp     word [cs:_$P_RC],_$P_Syntax ;AC034; if error, examine the next type
00001D2A 750F                         jne     short _$P_Match_Exit  ;AN000;
                                  _$P_Match07:                              ;AN000;
                                      ;test   ax,_$P_Simple_S       ; 2000h;AN000; Simple string
                                      ; 07/07/2023
00001D2C F6C420                       test    ah,(_$P_Simple_S>>8) ; 20h
00001D2F 740A                         jz      short _$P_Match09     ;AN000;

00001D31 2EC706[7119]0000             mov     word [cs:_$P_RC],_$P_No_Error ;AC034; assume no error
00001D38 E8BA01                       call    _$P_Simple_String     ;AN000; do process
                                  _$P_Match09:                              ;AN000;
                                  _$P_Match_Exit:                           ;AN000;
00001D3B 2E833E[141A]01               cmp     word [cs:_$P_err_flag],_$P_error_filespec ;AC034; bad filespec ?
00001D41 750F                         jne     short _$P_Match2_Exit ;AN033; no, continue
00001D43 2E833E[7119]00               cmp     word [cs:_$P_RC],_$P_No_Error ;AN033;AC034;; check for other errors ?
00001D49 7507                         jne     short _$P_Match2_Exit ;AN033; no, continue
00001D4B 2EC706[7119]0900             mov     word [cs:_$P_RC],_$P_Syntax ;AN033;AC034;; set error flag
                                  _$P_Match2_Exit:                  ;AN033;
00001D52 58                           pop     ax                    ;AN000;
00001D53 C3                           retn                          ;AN000;

                                  ;******************************************************************************
                                  ; _$P_Remove_Colon;
                                  ;
                                  ; Function: Remove colon at end
                                  ;
                                  ; Input:    cs:SI points to string buffer to be examineed
                                  ;
                                  ; Output:   None
                                  ;
                                  ; Use:      _$P_Chk_DBCS
                                  ;******************************************************************************

                                  _$P_Remove_Colon:
00001D54 50                           push    ax                    ;AN000;
00001D55 56                           push    si                    ;AN000;
                                  _$P_RCOL_Loop:                            ;AN000;
00001D56 2E8A04                       mov     al,[cs:si]            ;AN000; get character
00001D59 08C0                         or      al,al                 ;AN000; end of string ?
00001D5B 741A                         jz      short _$P_RCOL_Exit   ;AN000; if yes, just exit

00001D5D 3C3A                         cmp     al,_$P_Colon          ;AN000; is it colon ?
00001D5F 750D                         jne     short _$P_RCOL00      ;AN000;

00001D61 2E807C0100                   cmp     byte [cs:si+1],_$P_NULL ;AN000; if so, next is NULL ?
00001D66 7506                         jne     short _$P_RCOL00      ;AN000; no, then next char

00001D68 2EC60400                     mov     byte [cs:si],_$P_NULL ;AN000; yes, remove colon
00001D6C EB09                         jmp     short _$P_RCOL_Exit   ;AN000; and exit.

                                  _$P_RCOL00:                       ;AN000;
00001D6E E80E04                       call    _$P_Chk_DBCS          ;AN000; if not colon, then check if
00001D71 7301                         jnc     short _$P_RCOL01      ;AN000; DBCS leading byte.

00001D73 46                           inc     si                    ;AN000; if yes, skip trailing byte
                                  _$P_RCOL01:                       ;AN000;
00001D74 46                           inc     si                    ;AN000; si points to next byte
00001D75 EBDF                         jmp     short _$P_RCOL_Loop   ;AN000; loop until NULL encountered

                                  _$P_RCOL_Exit:                            ;AN000;
00001D77 5E                           pop     si                    ;AN000;
00001D78 58                           pop     ax                    ;AN000;
00001D79 C3                           retn                          ;AN000;

                                  ;******************************************************************************
                                  ; _$P_Do_CAPS_String;
                                  ;
                                  ; Function: Perform capitalization along with the file case map table
                                  ;       or character case map table.
                                  ;
                                  ; Input:    AL = 2 : Use character table
                                  ;       AL = 4 : Use file table
                                  ;       cs:SI points to string buffer to be capitalized
                                  ;
                                  ; Output:   None
                                  ;
                                  ; Use:      _$P_Do_CAPS_Char, _$P_Chk_DBCS
                                  ;******************************************************************************

                                  _$P_Do_CAPS_String:
00001D7A 56                           push    si                    ;AN000;
00001D7B 52                           push    dx                    ;AN000;
00001D7C 88C2                         mov     dl,al                 ;AN000; save info id

                                  _$P_DCS_Loop:                             ;AN000;
00001D7E 2E8A04                       mov     al,[cs:si]            ;AN000; load charater and
00001D81 E8FB03                       call    _$P_Chk_DBCS          ;AN000; check if DBCS leading byte
00001D84 720C                         jc      short _$P_DCS00               ;AN000; if yes, do not need CAPS

00001D86 08C0                         or      al,al                 ;AN000; end of string ?
00001D88 740C                         jz      short _$P_DCS_Exit    ;AN000; then exit.

00001D8A E80C00                       call    _$P_Do_CAPS_Char      ;AN000; Here a SBCS char need to be CAPS
00001D8D 2E8804                       mov     [cs:si],al            ;AN000; stored upper case char to buffer
00001D90 EB01                         jmp     short _$P_DCS01               ;AN000; process next
                                  _$P_DCS00:                        ;AN000;
00001D92 46                           inc     si                    ;AN000; skip DBCS leading and trailing byte
                                  _$P_DCS01:                        ;AN000;
00001D93 46                           inc     si                    ;AN000; si point to next byte
00001D94 EBE8                         jmp     short _$P_DCS_Loop    ;AN000; loop until NULL encountered
                                  _$P_DCS_Exit:                             ;AN000;
00001D96 5A                           pop     dx                    ;AN000;
00001D97 5E                           pop     si                    ;AN000;
00001D98 C3                           retn

                                  ;******************************************************************************
                                  ; _$P_Do_CAPS_Char;
                                  ;
                                  ; Function: Perform capitalization along with the file case map table
                                  ;       or character case map table.
                                  ;
                                  ; Input:    DL = 2 : Use character table
                                  ;       DL = 4 : Use file table
                                  ;       AL = character to be capitalized
                                  ;
```

```
                                ; Output:   None
                                ;
                                ; Use:      INT 21h /w AH=65h
                                ;*************************************************************************
                                
                                _$P_Do_CAPS_Char:
00001D99 3C80                       cmp     al,_$P_ASCII80 ;80h    ;AN000; need upper case table ?
00001D9B 730B                       jae     short _$P_DCC_Go       ;AN000;
                                
00001D9D 3C61                       cmp     al,"a"                   ;AN000; if no,
00001D9F 7234                       jb      short _$P_CAPS_Ret     ;AN000;   check if  "a" <= AL <= "z"
                                
00001DA1 3C7A                       cmp     al,"z"                   ;AN000;
00001DA3 7730                       ja      short _$P_CAPS_Ret     ;AN000;   if yes, make CAPS
                                
00001DA5 24DF                       and     al,_$P_Make_Upper ;0DFh ;AN000;   else do nothing.
                                    ;jmp    short _$P_CAPS_Ret     ;AN000;
                                    ; 07/07/2023
00001DA7 C3                         retn
                                
                                _$P_DCC_Go:                          ;AN000;
00001DA8 53                         push    bx                       ;AN000;
00001DA9 06                         push    es                       ;AN000;
00001DAA 57                         push    di                       ;AN000;
                                
                                    ;;lea   di,[cs:_$P_Char_CAP_Ptr] ;AC034; or use char CAPS table ?
                                    ;lea    di,[_$P_Char_CAP_Ptr]
                                    ; 07/09/2023
00001DAB BF[061A]                   mov     di,_$P_Char_CAP_Ptr
                                _$P_DCC00:                           ;AN000;
00001DAE 2E3815                     cmp     [cs:di],dl               ;AN000; already got table address ?
00001DB1 7415                       je      short _$P_DCC01                ;AN000; if no,
                                
                                ;In this next section, ES will be used to pass a 5 byte workarea to INT 21h,
                                ; the GET COUNTYRY INFO call. This usage of ES is required by the function
                                ; call, regardless of what base register is currently be defined as cs.
                                
00001DB3 50                         push    ax                       ;AN000; get CAPS table thru DOS call
00001DB4 51                         push    cx                       ;AN000;
00001DB5 52                         push    dx                       ;AN000;
                                
00001DB6 0E                         push    cs                       ;AC036; pass current base seg into
                                                                     ;(Note: this used to push CS.  BUG...
00001DB7 07                         pop     es                       ;AN000;   ES reg, required for
                                                                     ;get extended country information
                                    ;mov    al,dl ; function         ;AN000; upper case table
                                    ; 07/07/2023
00001DB8 92                         xchg    ax,dx
00001DB9 B465                       mov     ah,_$P_DOS_Get_TBL ; 65h ;AN000; get extended CDI
00001DBB BBFFFF                     mov     bx,_$P_DOSTBL_Def ; -1;AN000; get active CON
00001DBE B90500                     mov     cx,_$P_DOSTBL_BL ; 5   ;AN000; buffer length
                                    ;mov    dx,_$P_DOSTBL_Def      ;AN000; get for default code page
                                    ; 07/07/2023
00001DC1 89DA                       mov     dx,bx ; 0FFFFh
                                                                     ;DI already set to point to buffer
00001DC3 CD21                       int     21h                      ;AN000; es:di point to buffer that
                                                                     ;now has been filled in with info
00001DC5 5A                         pop     dx                       ;AN000;
00001DC6 59                         pop     cx                       ;AN000;
00001DC7 58                         pop     ax                       ;AN000;
                                
                                _$P_DCC01:                           ;AN000;
                                
                                ;In this next section, ES will be used as the base of the XLAT table, provided
                                ; by the previous GET COUNTRY INFO DOS call.  This usage of ES is made
                                ; regardless of which base reg is currently the cs reg.
                                
                                    ; 14/04/2024
                                    ;mov    bx,[cs:di+_$P_DOS_TBL.Off] ;AN000; get offset of table
                                    ;mov    es,[cs:di+_$P_DOS_TBL.Seg] ;AN000; get segment of table
                                    ; 07/07/2023
00001DC8 2EC45D01                   les     bx,[cs:di+_$P_DOS_TBL.Off]
00001DCC 43                         inc     bx                       ;AC035; add '2' to
00001DCD 43                         inc     bx                       ;AC035;  BX reg
                                                                     ;AN000; skip length field
00001DCE 2C80                       sub     al,_$P_ASCII80 ; 80h     ;AN000; make char to index
                                    ;xlat   es:[bx]                  ;AN000; perform case map
00001DD0 26                         es
00001DD1 D7                         xlat
00001DD2 5F                         pop     di                       ;AN000;
00001DD3 07                         pop     es                       ;AN000;
00001DD4 5B                         pop     bx                       ;AN000;
                                _$P_CAPS_Ret:                                ;AN000;
00001DD5 C3                         retn                             ;AN000;
                                
                                ;*************************************************************************
                                ; _$P_Value / _$P_SValue
                                ;
                                ; Function:  Make 32bit value from cs:SI and see value list
                                ;         and make result buffer.
                                ;         _$P_SValue is an entry point for the signed value
                                ;         and this will simply call _$P_Value after the handling
                                ;         of the sign character, "+" or "-"
                                ;
                                ; Input:     cs:SI -> _$P_STRING_BUF
                                ;         ES:BX -> CONTROL block
                                ;
                                ; Output:    None
                                ;
                                ; Use:      _$P_Fill_Result, _$P_Check_OVF
                                ;
                                ; Vars: _$P_RC(W), _$P_Flags(RW)
                                ;*************************************************************************
                                
                                    ; 26/10/2022 - Retro DOS v4.0
                                    ; (MSDOS 5.0 IO.SYS - SYSINIT:1B0Bh)
                                
                                    ; 07/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                    ; (MSDOS 6.21 IO.SYS - SYSINIT:1C46h)
                                _$P_SValue:                          ;AN000; when signed value here
00001DD6 50                         push    ax                       ;AN000;
00001DD7 2E800E[7D19]80             or      byte [cs:_$P_Flags2],_$P_Signed ;AC034; indicate a signed numeric
00001DDD 2E8026[7D19]FD             and     byte [cs:_$P_Flags2],0FFh-_$P_Neg ;AC034; assume positive value
                                    ;and    byte [cs:_$P_Flags2],~_$P_Neg ; 07/07/2023
00001DE3 2E8A04                     mov     al,[cs:si]               ;AN000; get sign
00001DE6 3C2B                       cmp     al,_$P_Plus              ;AN000; "+" ?
00001DE8 740A                       je      short _$P_SVal00         ;AN000;
                                
00001DEA 3C2D                       cmp     al,_$P_Minus             ;AN000; "-" ?
00001DEC 7507                       jne     short _$P_SVal01         ;AN000; else
                                
00001DEE 2E800E[7D19]02             or      byte [cs:_$P_Flags2],_$P_Neg ;AC034; set this is negative value
                                _$P_SVal00:                          ;AN000;
00001DF4 46                         inc     si                       ;AN000; skip sign char
```

```
31594                                   _$P_Sval01:                         ;AN000;
31595 00001DF5 E80200                       call    _$P_Value               ;AN000; and process value
31596 00001DF8 58                           pop     ax                      ;AN000;
31597 00001DF9 C3                           retn
31598
31599                                   ;*****************************************************************************
31600
31601                                       ; 07/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
31602                                       ; (MSDOS 6.21 IO.SYS - SYSINIT:1C6Ah)
31603
31604                                       ; 26/10/2022
31605                                   _$P_Value:                          ;AN000;
31606 00001DFA 50                           push    ax                      ;AN000;
31607 00001DFB 51                           push    cx                      ;AN000;
31608 00001DFC 52                           push    dx                      ;AN000;
31609 00001DFD 56                           push    si                      ;AN000;
31610 00001DFE 31C9                         xor     cx,cx                   ;AN000; cx = higher 16 bits
31611 00001E00 31D2                         xor     dx,dx                   ;AN000; dx = lower 16 bits
31612 00001E02 53                           push    bx                      ;AN000; save control pointer
31613                                   _$P_Value_Loop:                             ;AN000;
31614 00001E03 2E8A04                       mov     al,[cs:si]              ;AN000; get character
31615 00001E06 08C0                         or      al,al                   ;AN000; end of line ?
31616 00001E08 7438                         jz      short _$P_Value00       ;AN000;
31617
31618 00001E0A E8DC00                       call    _$P_0099                ;AN000; make asc(0..9) to bin(0..9)
31619 00001E0D 722F                         jc      short _$P_Value_Err0    ;AN000;
31620
31621 00001E0F 30E4                         xor     ah,ah                   ;AN000;
31622 00001E11 89C5                         mov     bp,ax                   ;AN000; save binary number
31623
31624                                   ; 08/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
31625                                   ; Ref: Disassembled PCDOS 7.1 IBMBIO.COM SYSINIT code
31626                                   ;                           Erdogan Tan - July 2023
31627                                   %if 0
31628                                       shl     dx,1                    ;AN000; to have 2*x
31629                                       rcl     cx,1                    ;AN000; shift left w/ carry
31630                                       call    _$P_Check_OVF           ;AN000; Overflow occurred ?
31631                                       jc      short _$P_Value_Err0    ;AN000; then error, exit
31632
31633                                       mov     bx,dx                   ;AN000; save low(2*x)
31634                                       mov     ax,cx                   ;AN000; save high(2*x)
31635                                       shl     dx,1                    ;AN000; to have 4*x
31636                                       rcl     cx,1                    ;AN000; shift left w/ carry
31637                                       call    _$P_Check_OVF           ;AN000; Overflow occurred ?
31638                                       jc      short _$P_Value_Err0    ;AN000; then error, exit
31639
31640                                       shl     dx,1                    ;AN000; to have 8*x
31641                                       rcl     cx,1                    ;AN000; shift left w/ carry
31642                                       call    _$P_Check_OVF           ;AN000; Overflow occurred ?
31643                                       jc      short _$P_Value_Err0    ;AN000; then error, exit
31644
31645                                       add     dx,bx                   ;AN000; now have 10*x
31646                                       adc     cx,ax                   ;AN000; 32bit ADD
31647                                       call    _$P_Check_OVF           ;AN000; Overflow occurred ?
31648                                       jc      short _$P_Value_Err0    ;AN000; then error, exit
31649
31650                                       add     dx,bp                   ;AN000; Add the current one degree decimal
31651                                       adc     cx,0                    ;AN000; if carry, add 1 to high 16bit
31652                                       call    _$P_Check_OVF           ;AN000; Overflow occurred ?
31653                                       jc      short _$P_Value_Err0    ;AN000; then error, exit
31654
31655                                       inc     si                      ;AN000; update pointer
31656                                       jmp     short _$P_Value_Loop    ;AN000; loop until NULL encountered
31657                                   ;_$P_Value_Err0:
31658                                   %endif
31659                                   ;****
31660                                   %if 1
31661                                   ; 08/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
31662                                       ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:2130h)
31663
31664                                       ; 14/04/2024 - Retro DOS v5.0
31665                                       ;xor     ah,ah
31666                                       ;mov     bp,ax                  ; save binary number
31667
31668 00001E13 E81C00                       call    _$P_Value_2x_OVF       ; multiply cx:dx by 2 and then check overflow
31669 00001E16 89D3                         mov     bx,dx                  ; ax:bx = 2*(cx:dx)
31670 00001E18 89C8                         mov     ax,cx
31671 00001E1A E81500                       call    _$P_Value_2x_OVF       ; multiply cx:dx by 2 and then check overflow
31672 00001E1D E81200                       call    _$P_Value_2x_OVF       ; multiply cx:dx by 2 and then check overflow
31673 00001E20 01DA                         add     dx,bx                  ; 8*(cx:dx)+2*(cx:dx) = 10*(cx:dx)
31674 00001E22 11C1                         adc     cx,ax
31675 00001E24 E80F00                       call    _$P_Value_Chk_Add_OVF
31676 00001E27 01EA                         add     dx,bp                  ; Add the current one degree decimal
31677                                                                      ; if carry, add 1 to high 16bit
31678 00001E29 83D100                       adc     cx,0
31679 00001E2C E80700                       call    _$P_Value_Chk_Add_OVF  ; Overflow occurred ?
31680                                                                      ; then error, exit (without return here)
31681 00001E2F 46                           inc     si                     ; update pointer
31682 00001E30 EBD1                         jmp     short _$P_Value_Loop
31683
31684                                   _$P_Value_2x_OVF:
31685 00001E32 D1E2                         shl     dx,1                   ; to have 2*x
31686 00001E34 D1D1                         rcl     cx,1                   ; shift left w/ carry
31687                                   _$P_Value_Chk_Add_OVF:
31688 00001E36 E89E00                       call    _$P_Check_OVF          ; check overflow (for the last shift or add)
31689 00001E39 7201                         jc      short _$P_Value_OVF
31690 00001E3B C3                           retn
31691                                   _$P_Value_OVF:
31692 00001E3C 44                           inc     sp                     ; skip "call" return address to the caller
31693 00001E3D 44                           inc     sp
31694
31695                                   ;_$P_Value_Err0:
31696                                   %endif
31697                                   ;****
31698
31699                                   _$P_Value_Err0:                             ;AN000;
31700 00001E3E 5B                           pop     bx                     ;AN000;
31701 00001E3F E98300                       jmp     _$P_Value_Err          ;AN000; Bridge
31702                                   ;
31703                                   _$P_Value00:                                ;AN000;
31704 00001E42 5B                           pop     bx                     ;AN000; restore control pointer
31705 00001E43 2EF606[7D19]02               test    byte [cs:_$P_Flags2],_$P_Neg ;AC034; here cx,dx = 32bit value
31706 00001E49 740A                         jz      short _$P_Value01      ;AN000; was it negative ?
31707
31708 00001E4B F7D1                         not     cx                     ;AN000; +
31709 00001E4D F7D2                         not     dx                     ;AN000; |- Make 2's complement
31710 00001E4F 83C201                       add     dx,1                   ;AN000; |
31711 00001E52 83D100                       adc     cx,0                   ;AN000; +
31712
31713                                   _$P_Value01:                                ;AN000; / nval = 0
31714 00001E55 268B7706                     mov     si,[es:bx+_$P_Control_Blk.Value_List] ;AN000; si points to value list
31715 00001E59 268A04                       mov     al,[es:si]             ;AN000; get nval
31716                                       ; 07/09/2023
31717                                       ;cmp     al,_$P_nval_None ; 0  ;AN000; no value list ?
```

```
                                    ;;*jne  short _$P_Value02     ;AN000;
                                    ;;* 07/07/2023
                                    ;je     short _$P_Value05
                                    ; 07/09/2023
00001E5C 08C0                       or      al,al
00001E5E 7459                       jz      short _$P_Value05 ; _$P_nval_None

                                    ;mov    al,_$P_Number         ;AN000; Set type
                                    ;mov    ah,_$P_No_Tag         ;AN000; No ITEM_TAG set
                                    ; 07/07/2023
                                    ;*mov   ax,(_$P_No_Tag<<8)|_$P_Number
                                    ;*jmp   short _$P_Value_Exit  ;AN000;

                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS, SYSINIT compatibility)
                                    ; (SYSINIT:1BA5h)
                                    ; 12/12/2022
                                    ;nop    ; db  90h

                                _$P_Value02:                            ;AN000; / nval = 1
                                ;IF Val1SW                              ;AN000;(Check if value list id #1 is supported)
                                ;(tm07) cmp al,_$P_nval_Range     ;AN000; have range list ?
                                ;(tm07) jne short _$P_Value03     ;AN000;

00001E60 46                         inc     si                      ;AN000;
00001E61 268A04                     mov     al,[es:si]              ;AN000; al = number of range

                                    ; 07/09/2023
                                    ;cmp    al,_$P_No_nrng        ;AN000; (tm07)
                                    ;je     short _$P_Value03     ;AN000; (tm07)
00001E64 08C0                       or      al,al
00001E66 745D                       jz      short _$P_Value03 ; _$P_No_nrng

00001E68 46                         inc     si                      ;AN000; si points to 1st item_tag
                                _$P_Val02_Loop:                         ;AN000;
00001E69 2EF606[7D19]80             test    byte [cs:_$P_Flags2],_$P_Signed ;AC034;
00001E6F 751E                       jnz     short _$P_Val02_Sign  ;AN000;

00001E71 263B4C03                   cmp     cx,[es:si+_$P_Val_List.Val_XH] ;AN000; comp cx with XH
00001E75 7234                       jb      short _$P_Val02_Next  ;AN000;
00001E77 7706                       ja      short _$P_Val_In      ;AN000;

00001E79 263B5401                   cmp     dx,[es:si+_$P_Val_List.Val_XL] ;AN000; comp dx with XL
00001E7D 722C                       jb      short _$P_Val02_Next  ;AN000;

                                _$P_Val_In:                             ;AN000;
00001E7F 263B4C07                   cmp     cx,[es:si+_$P_Val_List.Val_YH] ;AN000; comp cx with YH (tm01)
00001E83 7726                       ja      short _$P_Val02_Next  ;AN000;
00001E85 7237                       jb      short _$P_Val_Found   ;AN000;

00001E87 263B5405                   cmp     dx,[es:si+_$P_Val_List.Val_YL] ;AN000; comp dx with YL
00001E8B 771E                       ja      short _$P_Val02_Next  ;AN000;

00001E8D EB2F                       jmp     short _$P_Val_Found   ;AN000;

                                _$P_Val02_Sign:                                 ;AN000;
00001E8F 263B4C03                   cmp     cx,[es:si+_$P_Val_List.Val_XH]       ;AN000; comp cx with XH
00001E93 7C16                       jl      short _$P_Val02_Next  ;AN000;
00001E95 7F06                       jg      short _$P_SVal_In     ;AN000;

00001E97 263B5401                   cmp     dx,[es:si+_$P_Val_List.Val_XL]       ;AN000; comp dx with XL
00001E9B 7C0E                       jl      short _$P_Val02_Next  ;AN000;

                                _$P_SVal_In:                                    ;AN000;
00001E9D 263B4C07                   cmp     cx,[es:si+_$P_Val_List.Val_YH]       ;AN000; comp cx with YH
00001EA1 7F08                       jg      short _$P_Val02_Next  ;AN000;

00001EA3 7C19                       jl      short _$P_Val_Found   ;AN000;

00001EA5 263B5405                   cmp     dx,[es:si+_$P_Val_List.Val_YL]       ;AN000; comp dx with YL
                                    ;jg     short _$P_Val02_Next  ;AN000;
                                    ;jmp    short _$P_Val_Found   ;AN000;
                                    ; 07/07/2023
00001EA9 7E13                       jng     short _$P_Val_Found

                                _$P_Val02_Next:                                 ;AN000;
00001EAB 83C609                     add     si,_$P_Len_Range      ;AN000;
00001EAE FEC8                       dec     al                    ;AN000; loop nrng times in AL
00001EB0 75B7                       jne     short _$P_Val02_Loop  ;AN000;
                                                                  ; / Not found
00001EB2 2EC706[7119]0600           mov     word [cs:_$P_RC],_$P_Out_Of_Range ;AC034;
                                    ;mov    al,_$P_Number         ;AN000;
                                    ;mov    ah,_$P_No_Tag         ;AN000; No ITEM_TAG set
                                _$P_Value05:                ;* 07/07/2023
                                    ; 07/07/2023
00001EB9 B801FF                     mov     ax,(_$P_No_Tag<<8)|_$P_Number
00001EBC EB11                       jmp     short _$P_Value_Exit  ;AN000;

                                _$P_Val_Found:                          ;AN000;
00001EBE B001                       mov     al,_$P_Number         ;AN000;
00001EC0 268A24                     mov     ah,[es:si]            ;AN000; found ITEM_TAG set
00001EC3 EB0A                       jmp     short _$P_Value_Exit  ;AN000;

                                _$P_Value03:                            ;AN000; / nval = 2

                                ;IF Val2SW                              ;AN000;(Check if value list id #2 is supported)
                                ;;;;cmp     al,$P_nval_Value      ; have match list ? ASSUME nval=2,
                                ;;;;jne     $P_Value04            ; even if it is 3 or more.
                                ;(tm07) inc si                      ;AN000;
                                ;(tm07) mov al,es:[si]              ;AN000; al = nrng
                                ;   mov     ah,$P_Len_Range        ;AN000;
                                ;   mul     ah                     ;AN000;  Skip nrng field
                                ;   inc     ax                     ;AN000;
                                ;   add     si,ax                  ;AN000; si points to nnval
                                ;   mov     al,es:[si]             ;AN000; get nnval
                                ;   inc     si                     ;AN000; si points to 1st item_tag
                                ;$P_Val03_Loop:                            ;AN000;
                                ;   cmp     cx,es:[si+$P_Val_XH]   ;AN000; comp cx with XH
                                ;   jne     $P_Val03_Next          ;AN000;
                                ;
                                ;   cmp     dx,es:[si+$P_Val_XL]   ;AN000; comp dx with XL
                                ;   je      $P_Val_Found           ;AN000;
                                ;
                                ;$P_Val03_Next:                            ;AN000;
                                ;   add     si,$P_Len_Value        ;AN000; points to next value choice
                                ;   dec     al                     ;AN000; loop nval times in AL
                                ;   jne     $P_Val03_Loop          ;AN000;
                                ;                                  ;AN000; / Not found
                                ;   mov     psdata_seg:$P_RC,$P_Not_in_Val ;AC034;
                                ;   mov     al,$P_Number           ;AN000;
                                ;   mov     ah,$P_No_Tag           ;AN000; No ITEM_TAG set
                                ;   jmp     short $P_Value_Exit    ;AN000;
                                ;
                                ;ENDIF                                  ;AN000;(of Val2SW)
                                ;$P_Value04:
```

```
                                _$P_Value_Err:                          ;AN000;
00001EC5 2EC706[7119]0900           mov     word [cs:_$P_RC],_$P_Syntax ;AC034;
                                    ;mov    al,_$P_String           ;AN000; Set type
                                    ;mov    ah,_$P_No_Tag           ;AN000; No ITEM_TAG set
                                    ; 07/09/2023
                                    ; 07/07/2023
00001ECC B803FF                     mov     ax,(_$P_No_Tag<<8)|_$P_String
                                _$P_Value_Exit:                         ;AN000;
00001ECF E855FD                     call    _$P_Fill_Result             ;AN000;
00001ED2 5E                         pop     si                      ;AN000;
00001ED3 5A                         pop     dx                      ;AN000;
00001ED4 59                         pop     cx                      ;AN000;
00001ED5 58                         pop     ax                      ;AN000;
00001ED6 C3                         retn                            ;AN000;

                                ; 28/03/2019 - Retro DOS v4.0

                                ;//////////////////////////////////////////////////////////////////////////
                                ; _$P_Check_OVF
                                ;
                                ; Function:  Check if overflow is occurred with consideration of
                                ;        signed or un-signed numeric value
                                ;
                                ; Input:     Flag register
                                ;
                                ; Output:    CY = 1  :    Overflow
                                ;
                                ; Vars:     _$P_Flags(R)
                                ;//////////////////////////////////////////////////////////////////////////

                                    ; 26/10/2022
                                _$P_Check_OVF:
00001ED7 9C                         pushf                           ;AN000;
00001ED8 2EF606[7D19]02             test    byte [cs:_$P_Flags2],_$P_Neg ;AC034; is it negative value ?
00001EDE 7502                       jnz     short _$P_COVF              ;AN000; if no, check overflow

00001EE0 9D                         popf                            ;AN000; by the CY bit
00001EE1 C3                         retn                            ;AN000;

                                _$P_COVF:                               ;AN000;
00001EE2 9D                         popf                            ;AN000; else,
00001EE3 7002                       jo      short _$P_COVF00        ;AN000; check overflow by the OF

00001EE5 F8                         clc                             ;AN000; indicate it with CY bit
00001EE6 C3                         retn                            ;AN000; CY=0 means no overflow

                                _$P_COVF00:                             ;AN000;
00001EE7 F9                         stc                             ;AN000; and CY=1 means overflow
00001EE8 C3                         retn                            ;AN000;

                                ;//////////////////////////////////////////////////////////////////////////
                                ; _$P_0099;
                                ;
                                ; Function:  Make ASCII 0-9 to Binary 0-9
                                ;
                                ; Input:     AL = character code
                                ;
                                ; Output:    CY = 1 : AL is not number
                                ;        CY = 0 : AL contains binary value
                                ;//////////////////////////////////////////////////////////////////////////

                                ; 08/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                %if 0
                                _$P_0099:
                                    cmp     al,"0"                  ;AN000;
                                    ;jb     short _$P_0099Err       ;AN000; must be 0 =< al =< 9
                                    ; 12/12/2022
                                    jb      short _$P_0099Err2  ; cf=1

                                    cmp     al,"9"                  ;AN000;
                                    ja      short _$P_0099Err       ;AN000; must be 0 =< al =< 9

                                    sub     al,"0"                  ;AN000; make char -> bin
                                    ; 12/12/2022
                                    ; cf=0
                                    ;clc                            ;AN000; indicate no error
                                    retn                            ;AN000;

                                _$P_0099Err:                                ;AN000;
                                    stc                             ;AN000; indicate error
                                _$P_0099Err2: ; 12/12/2022
                                    retn                            ;AN000;
                                %endif

                                ; 08/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                %if 1
                                _$P_0099:
00001EE9 3C30                       cmp     al,"0"                  ; cmp al,30h
00001EEB 7207                       jb      short _$P_0099Err       ; must be 0 =< al =< 9
00001EED 3C3A                       cmp     al,"9"+1                ; cmp al,3Ah
00001EEF F5                         cmc                             ; cf=0 -> cf=1
00001EF0 7202                       jb      short _$P_0099Err
00001EF2 2C30                       sub     al,"0"  ; sub al,30h    ; make char -> bin
                                    ; cf=0
                                _$P_0099Err:        ; cf=1
00001EF4 C3                         retn
                                %endif

                                ;//////////////////////////////////////////////////////////////////////////
                                ; _$P_Simple_String
                                ;
                                ; Function:  See value list for the simple string
                                ;        and make result buffer.
                                ;
                                ; Input:     cs:SI -> _$P_STRING_BUF
                                ;        ES:BX -> CONTROL block
                                ;
                                ; Output:    None
                                ;
                                ; Use:      _$P_Fill_Result, _$P_String_Comp
                                ;
                                ; Vars: _$P_RC(W)
                                ;//////////////////////////////////////////////////////////////////////////

                                _$P_Simple_String:
00001EF5 50                         push    ax                      ;AN000;
00001EF6 53                         push    bx                      ;AN000;
00001EF7 52                         push    dx                      ;AN000;
00001EF8 57                         push    di                      ;AN000;
00001EF9 268B7F06                   mov     di,[es:bx+_$P_Control_Blk.Value_List] ;AN000; di points to value list
00001EFD 268A05                     mov     al,[es:di]              ;AN000; get nval
00001F00 08C0                       or      al,al                   ;AN000; no value list ?
00001F02 7504                       jnz     short _$P_Sim00                 ;AN000; then
```

```
                                
00001F04 B4FF                       mov     ah,_$P_No_Tag           ;AN000; No ITEM_TAG set
00001F06 EB4C                       jmp     short _$P_Sim_Exit      ;AN000; and set result buffer
                                
                                _$P_Sim00:                          ;AN000;
                                ;IF Val3SW+KeySW                    ;AN000;(Check if keyword or value list id #3 is supported)
00001F08 3C03                       cmp     al,_$P_nval_String      ;AN000; String choice list provided ?
00001F0A 753F                       jne     short _$P_Sim01                 ;AN000; if no, syntax error
                                
00001F0C 47                         inc     di                      ;AN000;
00001F0D 268A05                     mov     al,[es:di]              ;AN000; al = nrng
00001F10 B409                       mov     ah,_$P_Len_Range        ;AN000;
00001F12 F6E4                       mul     ah                      ;AN000; Skip nrng field
00001F14 40                         inc     ax                      ;AN000; ax = (nrng*9)+1
00001F15 01C7                       add     di,ax                   ;AN000; di points to nnval
00001F17 268A05                     mov     al,[es:di]              ;AN000; get nnval
00001F1A B405                       mov     ah,_$P_Len_Value        ;AN000;
00001F1C F6E4                       mul     ah                      ;AN000; Skip nnval field
00001F1E 40                         inc     ax                      ;AN000; ax = (nnval*5)+1
00001F1F 01C7                       add     di,ax                   ;AN000; di points to nstrval
00001F21 268A05                     mov     al,[es:di]              ;AN000; get nstrval c
00001F24 47                         inc     di                      ;AC035; add '2' to
00001F25 47                         inc     di                      ;AC035;  DI reg
                                                                    ;AN000; di points to 1st string in list
                                _$P_Sim_Loop:                               ;AN000;
00001F26 268B2D                     mov     bp,[es:di]              ;AN000; get string pointer
00001F29 E83200                     call    _$P_String_Comp                 ;AN000; compare it with operand
00001F2C 7312                       jnc     short _$P_Sim_Found     ;AN000; found on list ?
                                
00001F2E 83C703                     add     di,_$P_Len_String ; 3   ;AN000; if no, point to next choice
00001F31 FEC8                       dec     al                      ;AN000; loop nstval times in AL
00001F33 75F1                       jne     short _$P_Sim_Loop      ;AN000;
                                                                    ;AN000; / Not found
00001F35 2EC706[7119]0800           mov     word [cs:_$P_RC],_$P_Not_In_Str ;AC034;
00001F3C B4FF                       mov     ah,_$P_No_Tag           ;AN000; No ITEM_TAG set
00001F3E EB14                       jmp     short _$P_Sim_Exit      ;AN000;
                                
                                _$P_Sim_Found:                              ;AN000;
00001F40 268A65FF                   mov     ah,[es:di-1]            ;AN000; set item_tag
00001F44 B002                       mov     al,_$P_List_Idx                 ;AN000;
00001F46 268B15                     mov     dx,[es:di]              ;AN000; get address of STRING
00001F49 EB0B                       jmp     short _$P_Sim_Exit0     ;AN000;
                                ;ENDIF                                      ;AN000;(of Val3SW+KeySW)
                                _$P_Sim01:                          ;AN000;
00001F4B 2EC706[7119]0900           mov     word [cs:_$P_RC],_$P_Syntax ;AC034;
00001F52 B4FF                       mov     ah,_$P_No_Tag           ;AN000; No ITEM_TAG set
                                _$P_Sim_Exit:                               ;AN000;
00001F54 B003                       mov     al,_$P_String           ;AN000; Set type
                                _$P_Sim_Exit0:                              ;AN000;
00001F56 E8CEFC                     call    _$P_Fill_Result                 ;AN000;
00001F59 5F                         pop     di                      ;AN000;
00001F5A 5A                         pop     dx                      ;AN000;
00001F5B 5B                         pop     bx                      ;AN000;
00001F5C 58                         pop     ax                      ;AN000;
00001F5D C3                         retn                            ;AN000;
                                
                                ;******************************************************************************
                                ; _$P_String_Comp:
                                ;
                                ; Function:  Compare two string
                                ;
                                ; Input:     cs:SI -> 1st string
                                ;        ES:BP -> 2nd string  (Must be upper case)
                                ;        ES:BX -> CONTROL block
                                ;
                                ; Output:    CY = 1 if not match
                                ;
                                ; Use:      _$P_Chk_DBCS, _$P_Do_CAPS_Char
                                ;
                                ; Vars: _$P_KEYor_SW_Ptr(W), _$P_Flags(R). _$P_KEYorSW_Ptr
                                ;******************************************************************************
                                
                                _$P_String_Comp:
00001F5E 50                         push    ax                      ;AN000;
00001F5F 55                         push    bp                      ;AN000;
00001F60 52                         push    dx                      ;AN000;
00001F61 56                         push    si                      ;AN000;
00001F62 B202                       mov     dl,_$P_DOSTBL_Char      ;AN000; use character case map table
                                _$P_SCOM_Loop:                              ;AN000;
00001F64 2E8A04                     mov     al,[cs:si]              ;AN000; get command character
00001F67 E81502                     call    _$P_Chk_DBCS            ;AN000; DBCS ?
00001F6A 723A                       jc      short _$P_SCOM00        ;AN000; yes,DBCS
                                
00001F6C E82AFE                     call    _$P_Do_CAPS_Char        ;AN000; else, upper case map before comparison
                                ;IF KeySW+SwSW                              ;AN000;(Check if keyword or switch is supported)
00001F6F 2EF606[7D19]08             test    byte [cs:_$P_Flags2],_$P_Key_Cmp ;AC034; keyword search ?
00001F75 740D                       jz      short _$P_SCOM04        ;AN000;
                                
00001F77 3C3D                       cmp     al,_$P_Keyword          ;AN000; "=" is delimiter
00001F79 751F                       jne     short _$P_SCOM03        ;AN000; IF "=" on command line AND  (bp+1=> char after the "=" in synonym
list)
                                
00001F7B 26807E0100                 cmp     byte [es:bp+1],_$P_NULL ;AN021;  at end of keyword string in the control block THEN
00001F80 756D                       jne     short _$P_SCOM_Differ   ;AN021;
                                
00001F82 EB13                       jmp     short _$P_SCOM05        ;AN000; keyword found in synonym list
                                
                                _$P_SCOM04:                                 ;AN000;
00001F84 2EF606[7D19]10             test    byte [cs:_$P_Flags2],_$P_SW_Cmp ;AC034; switch search ?
00001F8A 740E                       jz      short _$P_SCOM03        ;AN000;
                                
00001F8C 3C3A                       cmp     al,_$P_Colon            ;AN000; ":" is delimiter, at end of switch on command line
00001F8E 750A                       jne     short _$P_SCOM03        ;AN000; continue compares
                                
                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;cmp    byte [es:bp+0],_$P_NULL
                                    ; 11/12/2022
00001F90 26807E0000                 cmp     byte [es:bp],_$P_NULL   ;AN021; IF at end of switch on command AND
00001F95 7558                       jne     short _$P_SCOM_Differ   ;AN021;   at end of switch string in the control block THEN
                                
                                _$P_SCOM05:                                 ;AN000;   found a match
00001F97 46                         inc     si                      ;AN000; si points to just after "=" or ":"
00001F98 EB58                       jmp     short _$P_SCOM_Same     ;AN000; exit
                                
                                _$P_SCOM03:                                 ;AN000;
                                ;ENDIF                                              ;AN000;(of KeySW+SwSW)
                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;cmp    al,[es:bp+0]
                                    ; 11/12/2022
00001F9A 263A4600                   cmp     al,[es:bp]              ;AN000; compare operand w/ a synonym
00001F9E 751B                       jne     short _$P_SCOM_Differ0          ;AN000; if different, check ignore colon option
                                
00001FA0 08C0                       or      al,al                   ;AN000; end of line
00001FA2 744E                       jz      short _$P_SCOM_Same     ;AN000; if so, exit
```

```
                                    ; 12/12/2022
                                    ;inc    si                      ;AN000; update operand pointer
                                    ;inc    bp                      ;AN000;    and synonym pointer
                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
00001FA4 EB11                       jmp     short _$P_SCOM01        ;AN000; loop until NULL or "=" or ":" found in case

                                _$P_SCOM00:                         ;AN000; Here al is DBCS leading byte
                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;cmp    al,[es:bp+0]
                                    ; 11/12/2022
00001FA6 263A4600                   cmp     al,[es:bp]              ;AN000; compare leading byte
00001FAA 7543                       jne     short _$P_SCOM_Differ ;AN000; if not match, say different

00001FAC 46                         inc     si                      ;AN000; else, load next byte
00001FAD 2E8A04                     mov     al,[cs:si]              ;AN000; and
00001FB0 45                         inc     bp                      ;AN000;
                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;cmp    al,[es:bp+0]
                                    ; 11/12/2022
00001FB1 263A4600                   cmp     al,[es:bp]              ;AN000; compare 2nd byte
00001FB5 7538                       jne     short _$P_SCOM_Differ ;AN000; if not match, say different, too

                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ; 12/12/2022
                                _$P_SCOM01:
00001FB7 46                         inc     si                      ;AN000; else update operand pointer
00001FB8 45                         inc     bp                      ;AN000;                 and synonym pointer
                                ;_$P_SCOM01:                                ;AN000;
00001FB9 EBA9                       jmp     short _$P_SCOM_Loop     ;AN000; loop until NULL or "=" or "/" found in case

                                _$P_SCOM_Differ0:                   ;AN000;
                                ;IF SwSW                            ;AN000;(tm10)
00001FBB 2EF606[7D19]40             test    byte [cs:_$P_Flags2],_$P_SW ;AC034;(tm10)
00001FC1 740E                       jz      short _$P_not_applicable ;AN000;(tm10)

                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;test   word [es:bx+_$P_Control_Blk.Function_Flag],_$P_colon_is_not_necessary ;AN000;(tm10)
                                    ; 12/12/2022
00001FC3 26F6470220                 test    byte [es:bx+_$P_Control_Blk.Function_Flag],_$P_colon_is_not_necessary
00001FC8 7407                       jz      short _$P_not_applicable ;AN000;(tm10)

                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;cmp    byte [es:bp+0],_$P_NULL
                                    ; 11/12/2022
00001FCA 26807E0000                 cmp     byte [es:bp],_$P_NULL ;AN000;(tm10)
                                ;(deleted ;AN025;) jne short _$P_not_applicable ;AN000;(tm10)
00001FCF 7421                       je      short _$P_SCOM_Same     ;AN025;(tm10)

                                _$P_not_applicable:                         ;AN000;(tm10)
                                ;ENDIF                                      ;AN000;(tm10)

                                    ;test   word [es:bx+_$P_Control_Blk.Match_Flag],_$P_Ig_Colon
                                                                    ;AN000; ignore colon option specified ?
                                    ;test   byte [es:bx+_$P_Control_Blk.Match_Flag],_$P_Ig_Colon
                                    ; 12/12/2022
00001FD1 26F60710                   test    byte [es:bx],_$P_Ig_Colon
                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;test   word [es:bx],_$P_Ig_Colon ; 10h
00001FD5 7418                       jz      short _$P_SCOM_Differ ;AN000; if no, say different.

00001FD7 3C3A                       cmp     al,_$P_Colon            ;AN000; End up with ":" and
00001FD9 7509                       jne     short _$P_SCOM02        ;AN000;    subseqently

                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;cmp    byte [es:bp+0],_$P_NULL
                                    ; 11/12/2022
00001FDB 26807E0000                 cmp     byte [es:bp],_$P_NULL ;AN000; NULL ?
00001FE0 750D                       jne     short _$P_SCOM_Differ ;AN000; if no, say different

00001FE2 EB0E                       jmp     short _$P_SCOM_Same     ;AN000; else, say same

                                _$P_SCOM02:                         ;AN000;
00001FE4 3C00                       cmp     al,_$P_NULL             ;AN000; end up NULL and :
00001FE6 7507                       jne     short _$P_SCOM_Differ ;AN000;

                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;cmp    byte [es:bp+0],_$P_Colon
                                    ; 11/12/2022
00001FE8 26807E003A                 cmp     byte [es:bp],_$P_Colon;AN000; if no, say different
00001FED 7403                       je      short _$P_SCOM_Same     ;AN000; else, say same

                                _$P_SCOM_Differ:                    ;AN000;
00001FEF F9                         stc                             ;AN000; indicate not found
00001FF0 EB05                       jmp     short _$P_SCOM_Exit     ;AN000;

                                _$P_SCOM_Same:                              ;AN000;
                                    ; 12/12/2022
                                    ; cf=0
00001FF2 2E8936[8019]               mov     [cs:_$P_KEYorSW_Ptr],si ;AC034; for later use by keyword or switch
                                    ; 12/12/2022
                                    ;clc                            ;AN000; indicate found
                                _$P_SCOM_Exit:                              ;AN000;
00001FF7 5E                         pop     si                      ;AN000;
00001FF8 5A                         pop     dx                      ;AN000;
00001FF9 5D                         pop     bp                      ;AN000;
00001FFA 58                         pop     ax                      ;AN000;
00001FFB C3                         retn

                                ; 30/03/2019

                                ;IF FileSW+DrvSW                    ;AN000;(Check if file spec or drive only is supported)

                                ;******************************************************************************
                                ; _$P_File_Format;
                                ;
                                ; Function:  Check if the input string is valid file spec format.
                                ;        And set the result buffer.
                                ;
                                ; Input:     cs:SI -> _$P_STRING_BUF
                                ;        ES:BX -> CONTROL block
                                ;
                                ; Output:    None
                                ;
                                ; Use:      _$P_Fill_Result, _$P_Chk_DBCS, _$P_FileSp_Chk
                                ;
                                ; Vars: _$P_RC(W), _$P_SI_Save(W), _$P_Terminator(W), _$P_SaveSI_Cmpx(R)
                                ;   _$P_SaveSI_Cmpx(R)
                                ;******************************************************************************

                                _$P_File_Format:
00001FFC 50                         push    ax                      ;AN000;
00001FFD 57                         push    di                      ;AN000;
00001FFE 56                         push    si                      ;AN000;
```

```
32213 00001FFF 2E8B3E[7E19]             mov     di,[cs:_$P_SaveSI_Cmpx]       ;AC034; get user buffer address
32214                                   _$P_FileF_Loop0:                      ;AN000; / skip special characters
32215 00002004 2E8A04                   mov     al,[cs:si]                    ;AN000; load character
32216 00002007 08C0                     or      al,al                         ;AN000; end of line ?
32217 00002009 7413                     jz      short _$P_FileF_Err           ;AN000; if yes, error exit
32218
32219 0000200B E85D00                   call    _$P_FileSp_Chk                ;AN000; else, check if file special character
32220 0000200E 7523                     jne     short _$P_FileF03             ;AN000; if yes,
32221
32222 00002010 2EC606[141A]01           mov     byte [cs:_$P_err_flag],_$P_error_filespec
32223                                                                         ;AN033;AC034;; set error flag- bad char.
32224 00002016 5E                       pop     si                            ;AN033;
32225 00002017 2EC60400                 mov     byte [cs:si],_$P_NULL         ;AN033;
32226 0000201B 5F                       pop     di                            ;AN033;
32227 0000201C EB3E                     jmp     short _$P_FileF02             ;AN033;
32228
32229                                   _$P_FileF_Err:                        ;AN000;
32230 0000201E 5E                       pop     si                            ;AN000;
32231 0000201F 2EC60400                 mov     byte [cs:si],_$P_NULL         ;AN000;
32232 00002023 5F                       pop     di                            ;AN000;
32233
32234                                   ;test   word [es:bx+_$P_Control_Blk.Match_Flag],_$P_Optional ;AN000; is it optional ?
32235                                   ;test   byte [es:bx+_$P_Control_Blk.Match_Flag],_$P_Optional
32236                                   ; 12/12/2022
32237 00002024 26F60701                 test    byte [es:bx],_$P_Optional
32238                                   ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
32239                                   ;test   word [es:bx],_$P_Optional
32240 00002028 7532                     jnz     short _$P_FileF02             ;AN000;
32241
32242 0000202A 2EC706[7119]0200         mov     word [cs:_$P_RC],_$P_Op_Missing ;AC034; 3/17/87
32243 00002031 EB29                     jmp     short _$P_FileF02             ;AN000;
32244
32245                                   _$P_FileF03:                          ;AN000;
32246 00002033 58                       pop     ax                            ;AN000; discard save si
32247 00002034 56                       push    si                            ;AN000; save new si
32248                                   _$P_FileF_Loop1:                      ;AN000;
32249 00002035 2E8A04                   mov     al,[cs:si]                    ;AN000; load character (not special char)
32250 00002038 08C0                     or      al,al                         ;AN000; end of line ?
32251 0000203A 741E                     jz      short _$P_FileF_RLT           ;AN000;
32252
32253 0000203C E82C00                   call    _$P_FileSp_Chk                ;AN000; File special character ?
32254 0000203F 740B                     je      short _$P_FileF00             ;AN000;
32255
32256 00002041 E83B01                   call    _$P_Chk_DBCS                  ;AN000; no, then DBCS ?
32257 00002044 7302                     jnc     short _$P_FileF01             ;AN000;
32258 00002046 47                       inc     di                            ;AN000; if yes, skip next byte
32259 00002047 46                       inc     si                            ;AN000;
32260                                   _$P_FileF01:                          ;AN000;
32261 00002048 47                       inc     di                            ;AN000;
32262 00002049 46                       inc     si                            ;AN000;
32263 0000204A EBE9                     jmp     short _$P_FileF_Loop1         ;AN000;
32264                                   ;
32265                                   _$P_FileF00:                          ;AN000;
32266 0000204C 2EA2[7719]               mov     [cs:_$P_Terminator],al;AC034;
32267 00002050 2EC60400                 mov     byte [cs:si],_$P_NULL         ;AN000; update end of string
32268 00002054 47                       inc     di                            ;AN000;
32269 00002055 2E893E[7319]             mov     [cs:_$P_SI_Save],di           ;AC034; update next pointer in command line
32270                                   _$P_FileF_RLT:                        ;AN000;
32271 0000205A 5E                       pop     si                            ;AN000;
32272 0000205B 5F                       pop     di                            ;AN000;
32273                                   _$P_FileF02:                          ;AN000;
32274 0000205C 58                       pop     ax                            ;AN000; (tm14)
32275                                   ;test   ax,_$P_File_Spc       ; 200h  ;AN000; (tm14)
32276                                   ; 08/07/2023
32277 0000205D F6C402                   test    ah,(_$P_File_Spc>>8) ; 2
32278 00002060 7408                     jz      short _$P_Drv_Only_Exit       ;AN000; (tm14)
32279
32280 00002062 50                       push    ax                            ;AN000; (tm14)
32281                                   ;mov    ah,_$P_No_Tag         ;AN000; set
32282                                   ;mov    al,_$P_File_Spec      ;AN000; result
32283                                   ; 08/07/2023
32284 00002063 B805FF                   mov     ax,(_$P_No_Tag<<8)|_$P_File_Spec ; 0FF05h
32285                                                                         ; set result
32286 00002066 E8BEFB                   call    _$P_Fill_Result               ;AN000; buffer to file spec
32287 00002069 58                       pop     ax                            ;AN000;
32288
32289                                   _$P_Drv_Only_Exit:                    ;AN000; (tm14)
32290 0000206A C3                       retn                                  ;AN000;
32291
32292                                   ;******************************************************************************
32293                                   ; _$P_FileSp_Chk
32294                                   ;
32295                                   ; Function:  Check if the input byte is one of file special characters
32296                                   ;
32297                                   ; Input:     cs:SI -> _$P_STRING_BUF
32298                                   ;        AL = character code to be examineed
32299                                   ;
32300                                   ; Output:    ZF = 1 , AL is one of special characters
32301                                   ;******************************************************************************
32302
32303                                   _$P_FileSp_Chk:
32304 0000206B 53                       push    bx                            ;AN000;
32305 0000206C 51                       push    cx                            ;AN000;
32306                                   ;;lea   bx,[cs:_$P_FileSp_Char] ;AC034; special character table
32307                                   ;lea    bx,[_$P_FileSp_Char]  ; "[]|<>+=;\"" at
32308                                                                 ; MSDOS 6.21 IO.SYS - SYSINIT:1838h
32309                                   ; 07/09/2023
32310 0000206D BB[0B1A]                 mov     bx,_$P_FileSp_Char
32311 00002070 B90900                   mov     cx,_$P_FileSp_Len ; 9 ;AN000; load length of it
32312                                   _$P_FileSp_Loop:                      ;AN000;
32313 00002073 2E3A07                   cmp     al,[cs:bx]                    ;AN000; is it one of special character ?
32314 00002076 7404                     je      short _$P_FileSp_Exit         ;AN000;
32315
32316 00002078 43                       inc     bx                            ;AN000;
32317 00002079 E2F8                     loop    _$P_FileSp_Loop                       ;AN000;
32318
32319 0000207B 41                       inc     cx                            ;AN000; reset ZF
32320                                   _$P_FileSp_Exit:                      ;AN000;
32321 0000207C 59                       pop     cx                            ;AN000;
32322 0000207D 5B                       pop     bx                            ;AN000;
32323 0000207E C3                       retn
32324
32325                                   ;ENDIF                                        ;AN000;(of FileSW+DrvSW)
32326
32327                                   ;IF DrvSW                             ;AN000;(Check if drive only is supported)
32328
32329                                   ;******************************************************************************
32330                                   ; _$P_Drive_Format;
32331                                   ;
32332                                   ; Function:  Check if the input string is valid drive only format.
32333                                   ;        And set the result buffer.
32334                                   ;
32335                                   ; Input:     cs:SI -> _$P_STRING_BUF
32336                                   ;        ES:BX -> CONTROL block
```

```
                                ;
                                ; Output:    None
                                ;
                                ; Use:      _$P_Fill_Result, _$P_Chk_DBCS
                                ;
                                ; Vars: _$P_RC(W)
                                ;******************************************************************************
                                _$P_Drive_Format:
0000207F 50                         push    ax                      ;AN000;
00002080 52                         push    dx                      ;AN000;
00002081 2E8A04                     mov     al,[cs:si]              ;AN000;
00002084 08C0                       or      al,al                   ;AN000; if null string
00002086 7436                       je      short _$P_Drv_Exit      ;AN000; do nothing

00002088 E8F400                     call    _$P_Chk_DBCS            ;AN000; is it leading byte ?
0000208B 722A                       jc      short _$P_Drv_Err       ;AN000;

0000208D 2E837C013A                 cmp     word [cs:si+1],_$P_Colon ;AN000; "d", ":", 0  ?
00002092 740D                       je      short _$P_DrvF00        ;AN000;

                                    ;test   word [es:bx+_$P_Control_Blk.Match_Flag],_$P_Ig_Colon
                                    ;test   byte [es:bx+_$P_Control_Blk.Match_Flag],_$P_Ig_Colon ;AN000; colon can be ignored?
                                    ; 12/12/2022
00002094 26F60710                   test    byte [es:bx],_$P_Ig_Colon
                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;test   word [es:bx],_$P_Ig_Colon
00002098 741D                       jz      short _$P_Drv_Err       ;AN000;

0000209A 2E807C0100                 cmp     byte [cs:si+1],_$P_NULL ;AN000; "d", 0  ?
0000209F 7516                       jne     short _$P_Drv_Err       ;AN000;

                                _$P_DrvF00:                         ;AN000;
000020A1 0C20                       or      al,_$P_Make_Lower       ;AN000; lower case
000020A3 3C61                       cmp     al,"a"                    ;AN000; drive letter must
000020A5 7210                       jb      short _$P_Drv_Err       ;AN000; in range of

000020A7 3C7A                       cmp     al,"z"                    ;AN000; "a"-"z"
000020A9 770C                       ja      short _$P_Drv_Err       ;AN000; if no, error

000020AB 2C60                       sub     al,"a"-1                  ;AN000; make text drive to binary drive
000020AD 88C2                       mov     dl,al                   ;AN000; set
                                    ;mov    ah,_$P_No_Tag           ;AN000; result
                                    ;mov    al,_$P_Drive            ;AN000; buffer
                                    ; 08/07/2023
000020AF B806FF                     mov     ax,(_$P_No_Tag<<8)|_$P_Drive ; 0FF06h
                                                                    ; set result buffer
000020B2 E872FB                     call    _$P_Fill_Result           ;AN000; to drive
000020B5 EB07                       jmp     short _$P_Drv_Exit      ;AN000;

                                _$P_Drv_Err:                        ;AN000;
000020B7 2EC706[7119]0900           mov     word [cs:_$P_RC],_$P_Syntax ;AC034;
                                _$P_Drv_Exit:                       ;AN000;
000020BE 5A                         pop     dx                      ;AN000;
000020BF 58                         pop     ax                      ;AN000;
000020C0 C3                         retn                            ;AN000;

                                ;ENDIF                              ;AN000;(of DrvSW)

                                ;******************************************************************************
                                ; _$P_Skip_Delim;
                                ;
                                ; Function: Skip delimiters specified in the PARMS list, white space
                                ;       and comma.
                                ;
                                ; Input:    DS:SI -> Command String
                                ;       ES:DI -> Parameter List
                                ;
                                ; Output:   CY = 1 if the end of line encounterd
                                ;       CY = 0 then SI move to 1st non-delimiter character
                                ;       AL = Last examineed character
                                ;
                                ; Use:          _$P_Chk_EOL, _$P_Chk_Delim,
                                ;
                                ; Vars:     _$P_Flags(R)
                                ;******************************************************************************

                                _$P_Skip_Delim:
                                _$P_Skip_Delim_Loop:                ;AN000;
000020C1 AC                         lodsb                           ;AN000;
000020C2 E81E00                     call    _$P_Chk_EOL             ;AN000; is it EOL character ?
000020C5 7416                       jz      short _$P_Skip_Delim_CY ;AN000; if yes, exit w/ CY on

000020C7 E84E00                     call    _$P_Chk_Delim           ;AN000; is it one of delimiters ?
000020CA 7514                       jnz     short _$P_Skip_Delim_NCY ;AN000; if no, exit w/ CY off

000020CC 2EF606[7D19]20             test    byte [cs:_$P_Flags2],_$P_Extra ;AC034; extra delim or comma found ?
000020D2 74ED                       jz      short _$P_Skip_Delim_Loop ;AN000; if no, loop

000020D4 2EF606[7D19]41             test    byte [cs:_$P_Flags2],_$P_SW+_$P_equ ;AC034; /x , or xxx=zzz , (tm08)
                                    ;jz     short _$P_Exit_At_Extra ;AN000; no switch, no keyword (tm08)
                                    ; 08/07/2023
                                    ; cf=0
000020DA 7505                       jnz     short _$P_Skip_Delim_Exit
000020DC C3                         retn

                                    ;dec    si                      ;AN000; backup si for next call (tm08)
                                    ;jmp    short _$P_Exit_At_Extra ;AN000; else exit w/ CY off
                                    ; 12/12/2022
                                    ; cf=0
                                    ; 08/07/2023
                                    ;jmp    short _$P_Skip_Delim_Exit

                                _$P_Skip_Delim_CY:                  ;AN000;
000020DD F9                         stc                             ;AN000; indicate EOL
000020DE EB01                       jmp     short _$P_Skip_Delim_Exit ;AN000;

                                _$P_Skip_Delim_NCY:                 ;AN000;
000020E0 F8                         clc                             ;AN000; indicate non delim
                                _$P_Skip_Delim_Exit:                ;AN000; in this case, need
000020E1 4E                         dec     si                      ;AN000;  backup index pointer
                                    ; 08/07/2023
                                    ; 12/12/2022
                                ;_$P_Exit_At_Extra:        ; cf=0
000020E2 C3                         retn                            ;AN000;

                                    ; 12/12/2022
                                ;_$P_Exit_At_Extra:                 ;AN000;
                                    ;clc                            ;AN000; indicate extra delim
                                    ;retn                           ;AN000;

                                ;******************************************************************************
                                ; _$P_Chk_EOL;
                                ;
```

```
                                ; Function: Check if AL is one of End of Line characters.
                                ;
                                ; Input:    AL = character code
                                ;       ES:DI -> Parameter List
                                ;
                                ; Output:   ZF = 1 if one of End of Line characters
                                ;*************************************************************************
                                _$P_Chk_EOL:
000020E3 53                         push    bx                      ;AN000;
000020E4 51                         push    cx                      ;AN000;
000020E5 3C0D                       cmp     al,_$P_CR               ;AN000; Carriage return ?
000020E7 742C                       je      short _$P_Chk_EOL_Exit        ;AN000;
000020E9 3C00                       cmp     al,_$P_NULL             ;AN000; zero ?
000020EB 7428                       je      short _$P_Chk_EOL_Exit        ;AN000;
                                ;IF LFEOLSW                         ;AN028; IF LF TO BE ACCEPTED AS EOL
000020ED 3C0A                       cmp     al,_$P_LF               ;AN000; Line feed ?
000020EF 7424                       je      short _$P_Chk_EOL_Exit        ;AN000;
                                ;ENDIF                              ;AN028;
000020F1 26807D0202                 cmp     byte [es:di+_$P_PARMS_Blk.Num_Extra],_$P_I_Have_EOL
                                                                    ;AN000; EOL character specified ?
000020F6 721D                       jb      short _$P_Chk_EOL_Exit        ;AN000;
000020F8 31DB                       xor     bx,bx                   ;AN000;
000020FA 268A5D03                   mov     bl,[es:di+_$P_PARMS_Blk.Len_Extra_Delim]
                                                                    ;AN000; get length of delimiter list
000020FE 83C304                     add     bx,_$P_Len_PARMS        ;AN000; skip it
                                    ; 08/07/2023
00002101 31C9                       xor     cx,cx ; *
00002103 26803900                   cmp     byte [es:bx+di],_$P_I_Use_Default ;AN000; No extra EOL character ?
00002107 740B                       je      short _$P_Chk_EOL_NZ    ;AN000;
                                    ; 08/07/2023
                                    ;;xor   cx,cx                   ;AN000; Get number of extra character
                                    ;xor    ch,ch ; *
00002109 268A09                     mov     cl,[es:bx+di]           ;AN000;
                                _$P_Chk_EOL_Loop:                   ;AN000;
0000210C 43                         inc     bx                      ;AN000;
0000210D 263A01                     cmp     al,[es:bx+di]           ;AN000; Check extra EOL character
00002110 7403                       je      short _$P_Chk_EOL_Exit        ;AN000;
00002112 E2F8                       loop    _$P_Chk_EOL_Loop        ;AN000;
                                    ; 08/07/2023
                                    ; cx=0
                                _$P_Chk_EOL_NZ:                     ;AN000;
                                    ;cmp    al,_$P_CR               ;AN000; reset ZF
                                    ; 08/07/2023
00002114 41                         inc     cx  ; zf=0  (cx=1) ; *
                                _$P_Chk_EOL_Exit:                   ;AN000;
00002115 59                         pop     cx                      ;AN000;
00002116 5B                         pop     bx                      ;AN000;
00002117 C3                         retn

                                ;*************************************************************************
                                ;
                                ; _$P_Chk_Delim;
                                ;
                                ; Function: Check if AL is one of delimiter characters.
                                ;       if AL+[si] is DBCS blank, it is replaced with two SBCS
                                ;       blanks.
                                ;
                                ; Input:    AL = character code
                                ;       DS:SI -> Next Character
                                ;       ES:DI -> Parameter List
                                ;
                                ; Output:   ZF = 1 if one of delimiter characters
                                ;       SI points to the next character
                                ; Vars:  _$P_Terminator(W), _$P_Flags(W)
                                ;*************************************************************************

                                    ; 08/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                    ; MSDOS 6.21 IO.SYS - SYSINIT:1FAEh
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:2451h) ; (Retro DOS v5.0)

                                _$P_Chk_Delim:
00002118 53                         push    bx                      ;AN000;
00002119 51                         push    cx                      ;AN000;
0000211A 2EC606[7719]20             mov     byte [cs:_$P_Terminator],_$P_Space
                                                                    ;AC034; Assume terminated by space
                                    ;and    byte [cs:_$P_Flags20,0DFh
00002120 2E8026[7D19]DF             and     byte [cs:_$P_Flags2],0FFh-_$P_Extra ;AC034;
00002126 3C20                       cmp     al,_$P_Space ; 20h      ;AN000; Space ?
00002128 7423                       je      short _$P_Chk_Delim_Exit ;AN000;

0000212A 3C09                       cmp     al,_$P_TAB              ;AN000; TAB ?
0000212C 741F                       je      short _$P_Chk_Delim_Exit ;AN000;

0000212E 3C2C                       cmp     al,_$P_Comma            ;AN000; Comma ?
00002130 741E                       je      short _$P_Chk_Delim_Exit0 ;AN000;

                                ; Note: _$P_Chk_Delim00 part of code is nonsense here
                                ;        because _$P_Space = _$P_DBSP1 = 20h
                                ;       Erdogan Tan - 08/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                ;_$P_Chk_Delim00:
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:246Bh)
                                    ; (MSDOS 6.21 IO.SYS - SYSINIT:1FC8h)
                                %if 0
                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                _$P_Chk_Delim00:                    ;AN000;
                                    cmp     al,_$P_DBSP1   ; 20h    ;AN000; 1st byte of DBCS Space ?
                                    jne     short _$P_Chk_Delim01 ;AN000;

                                    cmp     byte [si],_$P_DBSP2 ; 20h ;AN000; 2nd byte of DBCS Space ?
                                    jne     short _$P_Chk_Delim01 ;AN000;

                                    mov     al,_$P_Space            ;AN000;
                                    inc     si                      ;AN000; make si point to next character
                                    cmp     al,al                   ;AN000; Set ZF
                                    jmp     short _$P_Chk_Delim_Exit ;AN000;
                                %endif

                                _$P_Chk_Delim01:                    ;AN000;
00002132 26807DFE01                 cmp     byte [es:di-_$P_PARMS_Blk.Num_Extra],_$P_I_Have_Delim
                                                                    ;AN000; delimiter character specified ?
00002137 7214                       jb      short _$P_Chk_Delim_Exit ;AN000;

                                    ;xor    cx,cx                   ;AN000;
00002139 30ED                       xor     ch,ch
                                    ;mov    cl,[es:di+3]
0000213B 268A4D03                   mov     cl,[es:di+_$P_PARMS_Blk.Len_Extra_Delim]
                                                                    ;AN000; get length of delimiter list
                                    ;or     cx,cx                   ;AN000; No extra Delim character ?
                                    ;jz     short _$P_Chk_Delim_NZ        ;AN000;
                                    ; 08/07/2023
0000213F E30B                       jcxz    _$P_Chk_Delim_NZ

00002141 BB0300                     mov     bx,_$P_Len_PARMS-1 ; 3;AN000; set bx to 1st extra delimiter
                                _$P_Chk_Delim_Loop:                 ;AN000;
```

```
32585 00002144 43                       inc     bx                      ;AN000;
32586 00002145 263A01                   cmp     al,[es:bx+di]           ;AN000; Check extra Delim character
32587 00002148 7406                     je      short _$P_Chk_Delim_Exit0 ;AN000;
32588
32589 0000214A E2F8                     loop    _$P_Chk_Delim_Loop      ;AN000; examine all extra delimiter
32590
32591                                 _$P_Chk_Delim_NZ:                   ;AN000;
32592                                     ;cmp    al,_$P_Space            ;AN000; reset ZF
32593                                     ; 08/07/2023
32594                                     ; cx=0 here
32595 0000214C 41                       inc     cx ; cx=1, zf=0
32596                                 _$P_Chk_Delim_Exit:                         ;AN000;
32597                                 _$P_ChkDfin:                                ;AN000;
32598 0000214D 59                       pop     cx                      ;AN000;
32599 0000214E 5B                       pop     bx                      ;AN000;
32600 0000214F C3                       retn                            ;AN000;
32601
32602                                 _$P_Chk_Delim_Exit0:                        ;AN000;
32603 00002150 2EA2[7719]               mov     [cs:_$P_Terminator],al ;AC034; keep terminated delimiter
32604 00002154 2EF606[7D19]01           test    byte [cs:_$P_Flags2],_$P_equ ;AN027;AC034;; if terminating a key=
32605 0000215A 7506                     jnz     short _$P_No_Set_Extra      ;AN027; then do not set the EXTRA bit
32606
32607 0000215C 2E800E[7D19]20           or      byte [cs:_$P_Flags2],_$P_Extra
32608                                                                       ;AC034; flag terminated extra delimiter or comma
32609                                 _$P_No_Set_Extra:                   ;AN027;
32610 00002162 38C0                     cmp     al,al                   ;AN000; set ZF
32611 00002164 EBE7                     jmp     short _$P_Chk_Delim_Exit ;AN000;
32612
32613                                 ;****************************************************************************
32614                                 ; _$P_Chk_Switch;
32615                                 ;
32616                                 ; Function: Check if AL is the switch character not in first position of
32617                                 ;       _$P_STRING_BUF
32618                                 ;
32619                                 ; Input:    AL = character code
32620                                 ;       BX = current pointer within _$P_String_Buf
32621                                 ;       SI =>next char on command line (following the one in AL)
32622                                 ;
32623                                 ; Output:   CF = 1 (set)if AL is switch character, and not in first
32624                                 ;            position, and has no chance of being part of a date string,
32625                                 ;            i.e. should be treated as a delimiter.
32626                                 ;
32627                                 ;       CF = 0 (reset, cleared) if AL is not a switch char, is in the first
32628                                 ;            position, or is a slash but may be part of a date string, i.e.
32629                                 ;            should not be treated as a delimiter.
32630                                 ;
32631                                 ; Vars:  _$P_Terminator(W)
32632
32633                                 ; Use:       _$P_0099
32634                                 ;****************************************************************************
32635
32636                                 _$P_Chk_Switch:
32637                                     ;;lea   bp,[cs:_$P_STRING_BUF];AN020;AC034
32638                                     ;lea    bp,[_$P_STRING_BUF]    ;BP=OFFSET of _$P_String_Buf even in group addressing
32639                                     ; 08/07/2023
32640 00002166 BD[8619]                 mov     bp,_$P_STRING_BUF
32641
32642                                 ;   .IF <BX NE BP> THEN             ;AN020;IF not first char THEN
32643 00002169 39EB                     cmp     bx,bp                   ;AN000;
32644 0000216B 7406                     je      short _$P_STRUC_L2      ;AN000;
32645
32646                                 ;   .IF <AL EQ _$P_Switch> THEN     ;AN020;otherwise see if a slash
32647 0000216D 3C2F                     cmp     al,_$P_Switch           ;AN000;
32648 0000216F 750C                     jne     short _$P_STRUC_L5      ;AN000;
32649
32650 00002171 F9                       stc                             ;AN020;not in first position and is slash
32651                                   ;jmp     short _$P_STRUC_L1     ;AN000;
32652                                   ; 12/12/2022
32653 00002172 C3                       retn
32654
32655                                 ; 12/12/2022
32656                                 ;_$P_STRUC_L5:                              ;AN000;
32657                                 ;   CLC                             ;AN020;not a slash
32658                                 ;;      .ENDIF                      ;AN020;
32659                                 ;;  .ELSE                           ;AN020;is first char in the buffer, ZF=0
32660                                 ;   jmp     short _$P_STRUC_L1      ;AN000;
32661
32662                                 _$P_STRUC_L2:                               ;AN000;
32663                                 ;   .IF <AL EQ _$P_Switch> THEN     ;AN020;
32664 00002173 3C2F                     cmp     al,_$P_Switch           ;AN000;
32665 00002175 7506                     jne     short _$P_STRUC_L12     ;AN000;
32666
32667 00002177 2E800E[7D19]40           or      byte [cs:_$P_Flags2],_$P_SW ;AN020 ;AC034;;could be valid switch, first char and is slash
32668                                 ;   .ENDIF                          ;AN020;
32669
32670                                     ; 12/12/2022
32671                                     ; cf=0
32672                                     ;retn
32673
32674                                 _$P_STRUC_L5:
32675                                     ; 12/12/2022
32676                                 _$P_STRUC_L12:                              ;AN000;
32677 0000217D F8                       clc                             ;AN020;CF=0 indicating first char
32678                                 ;   .ENDIF                          ;AN020;
32679                                 _$P_STRUC_L1:                               ;AN000;
32680 0000217E C3                       retn                            ;AN000;
32681
32682                                 ;****************************************************************************
32683                                 ; _$P_Chk_DBCS:
32684                                 ;
32685                                 ;  Function: Check if a specified byte is in ranges of the DBCS lead bytes
32686                                 ;
32687                                 ;  Input:
32688                                 ;     AL    = Code to be examineed
32689                                 ;
32690                                 ;  Output:
32691                                 ;     If CF is on then a lead byte of DBCS
32692                                 ;
32693                                 ; Use: INT 21h w/AH=63
32694                                 ;
32695                                 ; Vars:  _$P_DBCSEV_Seg(RW), _$P_DBCSEV_Off(RW)
32696                                 ;****************************************************************************
32697
32698                                 _$P_Chk_DBCS:
32699 0000217F 1E                       push    ds                      ;AN000;
32700 00002180 56                       push    si                      ;AN000;
32701 00002181 53                       push    bx                      ;AN000; (tm11)
32702                                   ;cmp    word [cs:_$P_DBCSEV_SEG],0 ;AC034; ALREADY SET ?
32703                                   ;jne    short _$P_DBCS00        ;AN000;
32704                                   ; 08/07/2023
32705 00002182 2E8B36[7A19]             mov     si,[cs:_$P_DBCSEV_SEG]
32706 00002187 21F6                     and     si,si ; 0 ?
32707 00002189 7525                     jnz     short _$P_DBCS00 ; already set
32708 0000218B 50                       push    ax                      ;AN000;
```

```
                                        push    ds                      ;AN000; (tm11)
                                        push    cx                      ;AN000;
                                        push    dx                      ;AN000;
                                        push    di                      ;AN000;
                                        push    bp                      ;AN000;
                                        push    es                      ;AN000;
                                        ; si = 0 ; 08/07/2023
                                        ;xor    si,si                   ;AN000;
000021A2 8EDE                           mov     ds,si ; 0               ;AN000;
000021A4 B80063                         mov     ax,_$P_DOS_GetEV ; 6300h ;AN000; GET DBCS EV CALL
000021A7 CD21                           int     21h                     ;AN000;
                                        ; DOS - 3.2+ only - GET DOUBLE BYTE CHARACTER SET LEAD TABLE
000021A9 8CDB                           mov     bx,ds                   ;AN000; (tm11)
000021AB 09DB                           or      bx,bx                   ;AN000; (tm11)
000021AD 07                             pop     es                      ;AN000;
000021AE 5D                             pop     bp                      ;AN000;
000021AF 5F                             pop     di                      ;AN000;
000021A0 5A                             pop     dx                      ;AN000;
000021A1 59                             pop     cx                      ;AN000;
000021A2 1F                             pop     ds                      ;AN000; (tm11)
000021A3 58                             pop     ax                      ;AN000;
000021A4 7424                           jz      short _$P_NON_DBCS      ;AN000;
                                _$P_DBCS02:                             ;AN000;
000021A6 2E8936[7819]                   mov     [cs:_$P_DBCSEV_OFF],si  ;AC034; save EV offset
000021AB 2E891E[7A19]                   mov     [cs:_$P_DBCSEV_SEG],bx  ;AC034; save EV segment (tm11)
                                _$P_DBCS00:                             ;AN000;
                                        ;mov    si,[cs:_$P_DBCSEV_OFF]  ;AC034; load EV offset
                                        ;mov    ds,[cs:_$P_DBCSEV_SEG]  ;AC034; and segment
                                        ; 08/07/2023
000021B0 2EC536[7819]                   lds     si,[cs:_$P_DBCSEV_OFF]
                                _$P_DBCS_LOOP:                                  ;AN000;
000021B5 833C00                         cmp     word [si],0             ;AN000; zero vector ?
000021B8 7410                           je      short _$P_NON_DBCS      ;AN000; then exit
000021BA 3A04                           cmp     al,[si]                 ;AN000;
000021BC 7208                           jb      short _$P_DBCS01        ;AN000; Check if AL is in
000021BE 3A4401                         cmp     al,[si+1]               ;AN000;   range of
000021C1 7703                           ja      short _$P_DBCS01        ;AN000;      the vector
000021C3 F9                             stc                             ;AN000; if yes, indicate DBCS and exit
000021C4 EB04                           jmp     short _$P_DBCS_EXIT     ;AN000;
                                _$P_DBCS01:                             ;AN000;
000021C6 46                             inc     si                      ;AC035; add '2' to
000021C7 46                             inc     si                      ;AC035;  SI reg
                                                                        ;AN000; get next vector
000021C8 EBEB                           jmp     short _$P_DBCS_LOOP     ;AN000; loop until zero vector found
                                _$P_NON_DBCS:                                   ;AN000;
                                        ; 12/12/2022
                                        ; cf=0
                                        ;clc                            ;AN000; indicate SBCS
                                _$P_DBCS_EXIT:                                  ;AN000;
000021CA 5B                             pop     bx                      ;AN000; (tm11)
000021CB 5E                             pop     si                      ;AN000;
000021CC 1F                             pop     ds                      ;AN000;
000021CD C3                             retn                            ;AN000;

                                ; SYSCONF.ASM - MSDOS 6.0 - 1991
                                ; ============================================================================
                                ; 27/03/2019 - Retro DOS v4.0

                                ;control block definitions for parser.
                                ;---------------------------------------------------------------------------
                                ; buffer = [n | n,m] {/e}

                                ; 30/03/2019

                                struc p_parms
00000000 ????                       resw    1       ; dw ?
00000002 ??                         resb    1       ; db 1  ; an extra delimiter list
00000003 ??                         resb    1       ; db 1 ; length is 1
00000004 ??                         resb    1       ; db ';' ; delimiter
                                .size:
                                endstruc

                                struc p_pos
00000000 ????                       resw    1       ; dw ?  ; numeric value??
00000002 ????                       resw    1       ; dw ?  ; function
00000004 ????                       resw    1       ; dw ?  ; result value buffer

                                ; note: by defining result_val before this structure, we could remove
                                ;  the "result_val" from every structure invocation

00000006 ????                       resw    1       ; dw ?  ; value list
00000008 ??                         resb    1       ; db 0  ; no switches/keywords
                                .size:
                                endstruc

                                struc       p_range
00000000 ??                         resb    1       ; db 1  ; range definition
00000001 ??                         resb    1       ; db 1  ; 1 definition of range
00000002 ??                         resb    1       ; db 1  ; item tag for this range
00000003 ????????                   resd    1       ; dd ?  ; numeric min
00000007 ????????                   resd    1       ; dd ?  ; numeric max
                                .size:
                                endstruc

                                ;---------------------------------------------------------------------------

                                    ; 26/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:1F48h)

                                    ; 08/07/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; MSDOS 6.21 IO.SYS - SYSINIT:2083h
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:251Dh) ; (Retro DOS v5.0)

                                ; buffer = [n | n,m] {/e}

                                ;buf_parms p_parms <buf_parmsx>
                                buf_parms:
000021CE [D321]                     dw      buf_parmsx
000021D0 01                         db      1       ; an extra delimiter list
000021D1 01                         db      1       ; length is 1
000021D2 3B                         db      ';'     ; delimiter

                                buf_parmsx:
000021D3 0102[DD21][F121]           dw      201h,buf_pos1,buf_pos2; min 1, max 2 positionals
000021D9 01                         db      1                     ; one switch
000021DA [0522]                     dw      sw_x_ctrl
000021DC 00                         db      0                     ; no keywords

                                ;buf_pos1 p_pos <8000h,0,result_val,buf_range_1>  ; numeric
                                buf_pos1:
000021DD 0080                       dw      8000h   ; numeric value??
000021DF 0000                       dw      0       ; function
000021E1 [1722]                     dw      result_val ; result value buffer
000021E3 [E621]                     dw      buf_range_1 ; value list
```

```
32833 000021E5 00                       db      0       ; no switches/keywords
32834
32835                                   ;buf_range_1 p_range <,,,1,99>          ; M050
32836                                   buf_range_1:
32837 000021E6 01                       db      1       ; range definition
32838 000021E7 01                       db      1       ; 1 definition of range
32839 000021E8 01                       db      1       ; item tag for this range
32840 000021E9 01000000                 dd      1       ; numeric min
32841 000021ED 63000000                 dd      99      ; numeric max
32842
32843                                   ;buf_pos2 p_pos <8001h,0,result_val,buf_range_2> ; optional num.
32844                                   buf_pos2:
32845 000021F1 0180                     dw      8001h
32846 000021F3 0000                     dw      0
32847 000021F5 [1722]                   dw      result_val
32848 000021F7 [FA21]                   dw      buf_range_2
32849 000021F9 00                       db      0
32850
32851                                   ;buf_range_2 p_range <,,,0,8>
32852                                   buf_range_2:
32853 000021FA 01                       db      1
32854 000021FB 01                       db      1
32855 000021FC 01                       db      1
32856 000021FD 00000000                 dd      0
32857 00002201 08000000                 dd      8
32858
32859                                   ;sw_x_ctrl p_pos <0,0,result_val,noval,1> ; followed by one switch
32860                                   sw_x_ctrl:
32861 00002205 0000                     dw      0
32862 00002207 0000                     dw      0
32863 00002209 [1722]                   dw      result_val
32864 0000220B [1622]                   dw      noval
32865 0000220D 01                       db      1       ; 1 switch
32866
32867                                   switch_x:
32868 0000220E 2F5800                   db      '/X',0          ; M016
32869
32870                                   p_buffers:
32871 00002211 0000                     dw      0       ; local variables
32872                                   p_h_buffers:
32873 00002213 0000                     dw      0
32874                                       ; 26/10/2022  (MSDOS 5.0 IO.SYS SYSINIT compatibility)
32875                                   p_buffer_slash_x:
32876 00002215 00                       db      0 ; 31/03/2019
32877
32878                                   ;-- common definitions ---------------------------------------------
32879
32880 00002216 00                       noval:      db      0
32881
32882                                   result_val:         ;label  byte
32883 00002217 00                       db      0               ; type returned
32884                                   result_val_itag:
32885 00002218 00                       db      0               ; item tag returned
32886                                   result_val_swoff:
32887 00002219 0000                     dw      0               ; es:offset of the switch defined
32888                                   rv_byte:    ;label  byte
32889 0000221B 00000000                 rv_dword: dd        0               ; value if number,or seg:offset to string.
32890
32891                                   ;-------------------------------------------------------------------------
32892
32893                                       ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
32894                                       ; (SYSINIT:1F99h)
32895
32896                                   ; break = [ on | off ]
32897
32898                                   ;brk_parms p_parms  <brk_parmsx>
32899                                   brk_parms:
32900 0000221F [2422]                   dw      brk_parmsx
32901 00002221 01                       db      1       ; an extra delimiter list
32902 00002222 01                       db      1       ; length is 1
32903 00002223 3B                       db      ';'     ; delimiter
32904
32905                                   brk_parmsx:
32906 00002224 0101[2A22]               dw      101h,brk_pos    ; min,max = 1 positional
32907 00002228 00                       db      0               ; no switches
32908 00002229 00                       db      0               ; no keywords
32909
32910                                   ;brk_pos p_pos <2000h,0,result_val,on_off_string> ; simple string
32911                                   brk_pos:
32912 0000222A 0020                     dw      2000h
32913 0000222C 0000                     dw      0
32914 0000222E [1722]                   dw      result_val
32915 00002230 [3322]                   dw      on_off_string
32916 00002232 00                       db      0
32917
32918                                   on_off_string:      ;label  byte
32919 00002233 03                       db      3               ; signals that there is a string choice
32920 00002234 00                       db      0               ; no range definition
32921 00002235 00                       db      0               ; no numeric values choice
32922 00002236 02                       db      2               ; 2 strings for choice
32923 00002237 01                       db      1               ; the 1st string tag
32924 00002238 [3D22]                   dw      on_string
32925 0000223A 02                       db      2               ; the 2nd string tag
32926 0000223B [4022]                   dw      off_string
32927
32928                                   on_string:
32929 0000223D 4F4E00                       db      "ON",0
32930                                   off_string:
32931 00002240 4F464600                 db      "OFF",0
32932
32933                                   p_ctrl_break:
32934 00002244 00                       db      0       ; local variable
32935
32936                                   ;-------------------------------------------------------------------------
32937
32938                                       ; 27/10/2022
32939
32940                                   ; country = n {m {path}}
32941                                   ; or
32942                                   ; country = n,,path
32943
32944                                   ;cntry_parms p_parms <cntry_parmsx>
32945                                   cntry_parms:
32946 00002245 [4A22]                   dw      cntry_parmsx
32947 00002247 01                       db      1
32948 00002248 01                       db      1
32949 00002249 3B                       db      ';'
32950
32951                                   cntry_parmsx:
32952 0000224A 0103[5422][6822]-        dw      301h,cntry_pos1,cntry_pos2,cntry_pos3 ; min 1, max 3 pos.
32952 00002250 [7122]
32953 00002252 00                       db      0               ; no switches
32954 00002253 00                       db      0               ; no keywords
32955
```

```
                                ;cntry_pos1 p_pos <8000h,0,result_val,cc_range> ; numeric value
                                cntry_pos1:
00002254 0080                       dw      8000h
00002256 0000                       dw      0
00002258 [1722]                     dw      result_val
0000225A [5D22]                     dw      cc_range
0000225C 00                         db      0

                                ;cc_range p_range <,,,1,999>
                                cc_range:
0000225D 01                         db      1
0000225E 01                         db      1
0000225F 01                         db      1
00002260 01000000                   dd      1
00002264 E7030000                   dd      999

                                ;cntry_pos2 p_pos <8001h,0,result_val,cc_range> ; optional num.
                                cntry_pos2:
00002268 0180                       dw      8001h
0000226A 0000                       dw      0
0000226C [1722]                     dw      result_val
0000226E [5D22]                     dw      cc_range
00002270 00                         db      0

                                ;cntry_pos3 p_pos <201h,0,result_val,noval>      ; optional filespec
                                cntry_pos3:
00002271 0102                       dw      201h
00002273 0000                       dw      0
00002275 [1722]                     dw      result_val
00002277 [1622]                     dw      noval
00002279 00                         db      0

                                p_cntry_code:
0000227A 0000                       dw      0       ; local variable
                                p_code_page:
0000227C 0000                       dw      0       ; local variable

                                ;--------------------------------------------------------------------------

                                    ; 27/10/2022

                                ; files = n

                                ;files_parms p_parms <files_parmsx>
                                files_parms:
0000227E [8322]                     dw      files_parmsx
00002280 01                         db      1
00002281 01                         db      1
00002282 3B                         db      ';'

                                files_parmsx:
00002283 0101[8922]                 dw      101h,files_pos ; min,max 1 positional
00002287 00                         db      0              ; no switches
00002288 00                         db      0              ; no keywords

                                ;files_pos p_pos <8000h,0,result_val,files_range,0> ; numeric value
                                files_pos:
00002289 0080                       dw      8000h
0000228B 0000                       dw      0
0000228D [1722]                     dw      result_val
0000228F [9222]                     dw      files_range
00002291 00                         db      0

                                ;files_range p_range <,,,8,255>
                                files_range:
00002292 01                         db      1
00002293 01                         db      1
00002294 01                         db      1
00002295 08000000                   dd      8
00002299 FF000000                   dd      255

                                p_files:
0000229D 00                         db      0              ; local variable

                                ;--------------------------------------------------------------------------

                                    ; 27/10/2022

                                ; fcbs = n,m

                                ;fcbs_parms p_parms <fcbs_parmsx>
                                fcbs_parms:
0000229E [A322]                     dw      fcbs_parmsx
000022A0 01                         db      1
000022A1 01                         db      1
000022A2 3B                         db      ';'

                                fcbs_parmsx:
000022A3 0102[AB22][BF22]           dw      201h,fcbs_pos_1,fcbs_pos_2 ; min,max = 2 positional
000022A9 00                         db      0                  ; no switches
000022AA 00                         db      0                  ; no keywords

                                ;fcbs_pos_1 p_pos <8000h,0,result_val,fcbs_range> ; numeric value
                                fcbs_pos_1:
000022AB 0080                       dw      8000h
000022AD 0000                       dw      0
000022AF [1722]                     dw      result_val
000022B1 [B422]                     dw      fcbs_range
000022B3 00                         db      0

                                ;fcbs_range p_range <,,,1,255>
                                fcbs_range:
000022B4 01                         db      1
000022B5 01                         db      1
000022B6 01                         db      1
000022B7 01000000                   dd      1
000022BB FF000000                   dd      255

                                ;fcbs_pos_2 p_pos <8000h,0,result_val,fcbs_keep_range> ; numeric value
                                fcbs_pos_2:
000022BF 0080                       dw      8000h
000022C1 0000                       dw      0
000022C3 [1722]                     dw      result_val
000022C5 [C822]                     dw      fcbs_keep_range
000022C7 00                         db      0

                                ;fcbs_keep_range p_range <,,,0,255>
                                fcbs_keep_range:
000022C8 01                         db      1
000022C9 01                         db      1
000022CA 01                         db      1
000022CB 00000000                   dd      0
000022CF FF000000                   dd      255
```

```
                                p_fcbs:     db      0               ; local variable
                                p_keep:     db      0               ; local variable

                                ;--------------------------------------------------------------------------

                                    ; 27/10/2022

                                ; lastdrive = x

                                ;ldrv_parms p_parms <ldrv_parmsx>
                                ldrv_parms:
                                    dw      ldrv_parmsx
                                    db      1
                                    db      1
                                    db      ';'

                                ldrv_parmsx:
                                    dw      101h,ldrv_pos  ; min,max = 1 positional
                                    db      0              ; no switches
                                    db      0              ; no keywords

                                ;ldrv_pos p_pos    <110h,10h,result_val,noval> ; drive only, ignore colon
                                ldrv_pos:                               ; remove colon at end
                                    dw      110h
                                    dw      10h
                                    dw      result_val
                                    dw      noval
                                    db      0

                                p_ldrv:     db      0               ; local variable

                                ;--------------------------------------------------------------------------

                                    ; 27/10/2022

                                ; stacks = n,m

                                ;stks_parms p_parms <stks_parmsx>
                                stks_parms:
                                    dw      stks_parmsx
                                    db      1
                                    db      1
                                    db      ';'

                                stks_parmsx:
                                    dw      202h,stks_pos_1,stks_pos_2 ; min,max = 2 positionals
                                    db      0                  ; no switches
                                    db      0                  ; no keywords

                                ;stks_pos_1 p_pos <8000h,0,result_val,stks_range> ; numeric value
                                stks_pos_1:
                                    dw      8000h
                                    dw      0
                                    dw      result_val
                                    dw      stks_range
                                    db      0

                                ;stks_range p_range <,,,0,64>
                                stks_range:
                                    db      1
                                    db      1
                                    db      1
                                    dd      0
                                    dd      64

                                ;stks_pos_2 p_pos <8000h,0,result_val,stk_size_range> ; numeric value
                                stks_pos_2:
                                    dw      8000h
                                    dw      0
                                    dw      result_val
                                    dw      stk_size_range
                                    db      0

                                ;stk_size_range p_range <,,,0,512>
                                stk_size_range:
                                    db      1
                                    db      1
                                    db      1
                                    dd      0
                                    dd      512

                                p_stack_count:
                                    dw      0       ; local variable
                                p_stack_size:
                                    dw      0       ; local variable

                                ;--------------------------------------------------------------------------

                                    ; 27/10/2022

                                ; multitrack = [ on | off ]

                                ;mtrk_parms p_parms <mtrk_parmsx>
                                mtrk_parms:
                                    dw      mtrk_parmsx
                                    db      1
                                    db      1
                                    db      ';'

                                mtrk_parmsx:
                                    dw      101h,mtrk_pos  ; min,max = 1 positional
                                    db      0              ; no switches
                                    db      0              ; no keywords

                                ;mtrk_pos p_pos <2000h,0,result_val,on_off_string> ; simple string
                                mtrk_pos:
                                    dw      2000h
                                    dw      0
                                    dw      result_val
                                    dw      on_off_string
                                    db      0

                                p_mtrk:     db      0               ; local variable

                                ;--------------------------------------------------------------------------

                                    ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:20B2h)

                                ; switches=/k

                                ;swit_parms p_parms <swit_parmsx>
                                swit_parms:
                                    dw      swit_parmsx
```

```
33204 0000233A 01                       db      1
33205 0000233B 01                       db      1
33206 0000233C 3B                       db      ';'
33207
33208                                   swit_parmsx:
33209 0000233D 0000                     dw      0               ; no positionals
33210                                   ; 08/07/2023 - Retro DOS v4.2 (MSDOS 6.21 IO.SYS, SYSINIT)
33211                                   ;db     5                 ; # of switches
33212                                   ; 14/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM, SYSINIT)
33213 0000233F 06                       db      6
33214                                   ; 27/10/2022 - Retro DOS v4.0 (MSDOS 5.0 IO.SYS, SYSINIT)
33215                                   ;db     3
33216 00002340 [4D23]                   dw      swit_k_ctrl     ; /k control
33217                                   ; 01/01/2023 - Retro DOS v4.2 ; *
33218 00002342 [5923]                   dw      swit_n_ctrl ; * ; /n control (for MULTI_CONFIG only)
33219 00002344 [6523]                   dw      swit_f_ctrl ; * ; /f control (for MULTI_CONFIG only)
33220 00002346 [7123]                   dw      swit_t_ctrl     ; /t control
33221 00002348 [7D23]                   dw      swit_w_ctrl     ; /w control
33222                                   ; 14/04/2024 - Retro DOS v5.0 ; **
33223 0000234A [8923]                   dw      swit_i_ctrl     ; /i control
33224 0000234C 00                       db      0               ; no keywords
33225
33226                                   ;swit_k_ctrl p_pos <0,0,result_val,noval,1> ; switch string follows
33227                                   swit_k_ctrl:
33228 0000234D 00000000[1722]-          dw      0,0,result_val,noval
33228 00002353 [1622]
33229 00002355 01                       db      1
33230 00002356 2F4B00                   swit_k:     db      '/K',0
33231
33232                                   ; 01/01/2023 - Retro DOS v4.2 (MSDOS 6.21 IO.SYS)
33233                                   ; (SYSINIT:220Ch) ; *
33234
33235                                   ; 27/10/2022 - Retro DOS v4.0 (MSDOS 5.0 IO.SYS, SYSINIT)
33236                                   ;
33237                                   ;swit_n_ctrl p_pos <0,0,result_val,noval,1> ; switch string follows
33238                                   swit_n_ctrl: ; *
33239 00002359 00000000[1722]-          dw      0,0,result_val,noval
33239 0000235F [1622]
33240 00002361 01                       db      1
33241 00002362 2F4E00                   swit_n: db  '/N',0
33242
33243                                   ;swit_f_ctrl p_pos <0,0,result_val,noval,1> ; switch string follows
33244                                   swit_f_ctrl: ; *
33245 00002365 00000000[1722]-          dw      0,0,result_val,noval
33245 0000236B [1622]
33246 0000236D 01                       db      1
33247 0000236E 2F4600                   swit_f: db  '/F',0
33248
33249                                       ; 27/10/2022
33250
33251                                   ;swit_t_ctrl p_pos <0,0,result_val,noval,1> ; switch string follows    M059
33252                                   swit_t_ctrl:
33253 00002371 00000000[1722]-          dw      0,0,result_val,noval
33253 00002377 [1622]
33254 00002379 01                       db      1
33255 0000237A 2F5400                   swit_t:     db      '/T',0                  ;                           M059
33256                                   ;swit_w_ctrl p_pos <0,0,result_val,noval,1> ; switch string follows    M063
33257                                   swit_w_ctrl:
33258 0000237D 00000000[1722]-          dw      0,0,result_val,noval
33258 00002383 [1622]
33259 00002385 01                       db      1
33260 00002386 2F5700                   swit_w:     db      '/W',0                  ;                           M063
33261
33262                                   ; 14/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM, SYSINIT)
33263                                   ;;;
33264                                   ;swit_i_ctrl p_pos <0,0,result_val,noval,1> ; switch string follows
33265                                   swit_i_ctrl:
33266 00002389 0000                     dw      0
33267 0000238B 0000                     dw      0
33268 0000238D [1722]                   dw      result_val
33269 0000238F [1622]                   dw      noval
33270 00002391 01                       db      1
33271 00002392 2F4900                   swit_i:     db      '/I',0
33272                                   ;;;
33273
33274                                   ;   There doesn't need to be p_swit_n or p_swit_f because /N and /F are
33275                                   ;   acted upon during MULTI_CONFIG processing; we only needed entries
33276                                   ;   in the above table to prevent the parsing code from complaining about them
33277
33278 00002395 00                     p_swit_k:   db      0       ; local variable
33279 00002396 00                     p_swit_t:   db      0       ; local variable                    M059
33280 00002397 00                     p_swit_w:   db      0       ; local variable                    M063
33281                                   ; 14/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
33282 00002398 00                     p_swit_i:   db      0
33283
33284                                   ;---------------------------------------------------------------------------
33285
33286                                       ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
33287                                       ; (SYSINIT:20E8h)
33288
33289                                   ; DOS = [ high | low ]
33290
33291                                   ;dos_parms p_parms  <dos_parmsx>
33292                                   dos_parms:
33293 00002399 [9E23]                       dw      dos_parmsx
33294 0000239B 01                           db      1
33295 0000239C 01                           db      1
33296 0000239D 3B                           db      ';'
33297                                   dos_parmsx:
33298 0000239E 01                           db      1               ; min parameters
33299 0000239F 02                           db      2               ; max parameters
33300 000023A0 [A623]                       dw      dos_pos         ;
33301 000023A2 [A623]                       dw      dos_pos         ;
33302 000023A4 00                           db      0               ; no switches
33303 000023A5 00                           db      0               ; no keywords
33304
33305                                   ;dos_pos p_pos     <2000h,0,result_val,dos_strings> ; simple string
33306                                   ;        p_pos     <2000h,0,result_val,dos_strings> ; simple string
33307                                   dos_pos:
33308 000023A6 00200000[1722]-              dw      2000h,0,result_val,dos_strings
33308 000023AC [B823]
33309 000023AE 00                           db      0
33310 000023AF 00200000[1722]-              dw      2000h,0,result_val,dos_strings
33310 000023B5 [B823]
33311 000023B7 00                           db      0
33312
33313                                   dos_strings:        ;label  byte
33314 000023B8 03                           db      3               ; signals that there is a string choice
33315 000023B9 00                           db      0               ; no range definition
33316 000023BA 00                           db      0               ; no numeric values choice
33317 000023BB 04                           db      4               ; 4 strings for choice
33318 000023BC 01                           db      1               ; the 1st string tag
33319 000023BD [E623]                       dw      hi_string
33320 000023BF 02                           db      2               ; the 2nd string tag
```

```
33321 000023C0 [EB23]                      dw      lo_string
33322 000023C2 03                          db      3
33323 000023C3 [EF23]                      dw      umb_string
33324 000023C5 04                          db      4
33325 000023C6 [F323]                      dw      noumb_string
33326
33327                                  ; 14/04/2024 - Retro DOS v5.0
33328                                  ; (PCDOS 7.1 IBMDOS.COM - SYSINIT:273Eh)
33329                                  ;;;
33330                                  dosdata_parms:
33331 000023C8 [CD23]                      dw      dosdata_parmsx ; DOSDATA = UMB|NOUMB
33332 000023CA 01                          db      1
33333 000023CB 01                          db      1
33334 000023CC 3B                          db      ';'
33335                                  dosdata_parmsx:
33336 000023CD 01                          db      1
33337 000023CE 01                          db      1               ; min,max = 1 positional
33338 000023CF [D323]                      dw      dosdata_pos
33339 000023D1 00                          db      0               ; no switches
33340 000023D2 00                          db      0               ; no keywords
33341
33342                                      ; dosdata_pos p_pos <2000h,0,result_val,dosdata_strings>
33343                                  dosdata_pos:
33344 000023D3 0020                        dw      2000h           ; simple string
33345 000023D5 0000                        dw      0
33346 000023D7 [1722]                      dw      result_val
33347 000023D9 [DC23]                      dw      dosdata_strings
33348 000023DB 00                          db      0
33349                                  dosdata_strings:
33350 000023DC 03                          db      3               ; signals that there is a string choice
33351 000023DD 00                          db      0               ; no range definition
33352 000023DE 00                          db      0               ; no numeric values choice
33353 000023DF 02                          db      2               ; 2 strings for choice
33354 000023E0 01                          db      1               ; the 1st string tag
33355 000023E1 [EF23]                      dw      umb_string      ; "UMB"
33356 000023E3 02                          db      2               ; the 2nd string tag
33357 000023E4 [F323]                      dw      noumb_string    ; "NOUMB"
33358                                  ;;;
33359
33360 000023E6 4849474800              hi_string:  db      "HIGH",0
33361 000023EB 4C4F5700                lo_string:  db      "LOW",0
33362 000023EF 554D4200                umb_string: db      "UMB",0
33363 000023F3 4E4F554D4200            noumb_string:       db      "NOUMB",0
33364
33365                                  p_dos_hi:
33366 000023F9 00                          db      0       ; local variable
33367                                                      ; BUGBUG : I dont know whether PARSER uses
33368                                                      ;          this variable or not
33369                                      ; 14/04/2024 (PCDOS 7.1 IBMBIO.COM)
33370 000023FA 00                          db      0
33371
33372                                  ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
33373                                  ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
33374                                  ;%if 0
33375
33376                                  ;************************************************************************ RICHID ****
33377
33378                                  ;include    highvar.inc    ; devicehigh variables (used by loadhigh also)
33379
33380                                  ; 30/03/2019 - Retro DOS v4.0
33381                                  ;--------------------------------------------------------------------------------
33382
33383                                  ;   Module:   HIGHVAR.INC - Data common to LOADHIGH and DEVICEHIGH, res seg
33384                                  ;
33385                                  ;   Date:     May 14, 1992
33386                                  ;
33387                                  ;*******************************************************************************
33388                                  ;
33389                                  ;   Modification log:
33390                                  ;
33391                                  ;     DATE    WHO      DESCRIPTION
33392                                  ;   --------  -------  -------------------------------------------------------
33393                                  ;   05/14/92  t-richj  Original
33394                                  ;   06/21/92  t-richj  Final revisions before check-in
33395                                  ;
33396                                  ;*******************************************************************************
33397                                  ;
33398                                  ; There are two primary definitions which need to be made, selectively, before
33399                                  ; this include file should be used.  These are:
33400                                  ;    HV_Extern - If this has been defined, variables for this module will be
33401                                  ;                declared as external.  Otherwise, variables will be declared
33402                                  ;                public, as well as defined, here.  LoadHigh declares HV_Extern
33403                                  ;                in stub.asm and loadhi.asm, and does not declare it in
33404                                  ;                rdata.asm... DeviceHigh does not declare HV_Extern anywhere
33405                                  ;                (as only one module, sysconf.asm, includes this file).
33406                                  ;    HV_LoadHigh - This should be defined when this module is going into
33407                                  ;                  command.com, for LoadHigh.  All of loadhi.asm, stub.asm and
33408                                  ;                  rdata.asm define this, while io.sys' sysconf.asm does not.
33409                                  ;
33410                                  ;*******************************************************************************
33411
33412                                  ; To keep track of which UMBs were specified on the DH/LH command lines, and
33413                                  ; to keep track of the minimum sizes given for each, there're two arrays kept
33414                                  ; in { IO.SYS: sysinitseg / COMMAND.COM: DATARES }... each is MAXUMB elements
33415                                  ; big.  16 should be around 14 too many for most users, so there's no expected
33416                                  ; space problem (it's just such a nice round number, eh?).
33417
33418                                  MAXUMB      equ     16
33419
33420                                  ; Memory elements owned by the system are marked as PSP address 8 in both the
33421                                  ; USA and Japan; Japanese systems also use 9 under more bizzarre conditions.
33422
33423                                  FreePSPOwner       equ     0       ; Free MCBs all have an owner PSP address of 0
33424                                  SystemPSPOwner     equ     8
33425                                  ;JapanPSPOwner     equ     9
33426
33427                                  ; for LoadHigh and DeviceHigh:
33428                                  ;
33429                                  ;   fInHigh - Is set to 1 during HideUMBs(), and back to zero in
33430                                  ;             UnHideUMBs().
33431                                  ;   fUmbTiny - Is set to 1 iff the user has specified /S on the command-
33432                                  ;              line.
33433                                  ;   SegLoad - Segment address for first UMB specified; set automatically.
33434                                  ;   UmbLoad - The load UMB number; for example, this is 3 if the user has
33435                                  ;             given a command-line like "/L:3,500;4"
33436                                  ;   UmbUsed - An array of characters, each of which is 1 iff the UMB
33437                                  ;             matching its index number was specified on the command-line;
33438                                  ;             for example, after "/L:3,500;4;7", UmbUsed[3], [4] and [7]
33439                                  ;             will be set to 1.  All others will be set to 0.
33440                                  ;   UmbSize - An array of words, each of which is interpereted as a size
33441                                  ;             specified by the user for a UMB (in the above example, all
33442                                  ;             elements would be zero save UmbSize[3], which would be 500.
33443                                  ;   fm_umb - Set to the old UMB link-state (0x80 or 0x00)
33444                                  ;   fm_strat - Set to the old memory-allocation strategy (0$00000???)
```

```
33445                                   ;   fm_argc  - Number of arguments received by ParseVar() (see ParseVar()
33446                                   ;               for details).
33447                                   
33448 000023FB 00                       fInHigh:  db       0
33449 000023FC 00                       fUmbTiny: db       0
33450 000023FD 0000                     SegLoad:  dw       0
33451 000023FF 00                       UmbLoad:  db       0
33452 00002400 00<rep 10h>              UmbUsed:  times MAXUMB db 0 ; times 16 db 0  ; db 16 dup(?)
33453 00002410 0000<rep 10h>            UmbSize:  times MAXUMB dw 0 ; times 16 dw 0  ; dw 16 dup(?)
33454 00002430 00                       fm_umb:   db       0
33455 00002431 00                       fm_strat: db       0
33456 00002432 00                       fm_argc:  db       0
33457                                   
33458                                   ; UmbLoad is set to UNSPECIFED, below, until /L:umb is read; at which point
33459                                   ; UmbLoad is set to the UMB number given.
33460                                   
33461                                   UNSPECIFIED equ    -1
33462                                   
33463                                   ;%endif ; 27/10/2022
33464                                   
33465                                   ;******************************************************************** RICHID ****
33466                                   
33467                                   ; 30/03/2019 - Retro DOS v4.0 (MSDOS 6.0, SYSCONF.ASM)
33468                                   ; ((MSDOS 6.21 IO.SYS -> SYNINIT:22BAh))
33469                                   
33470                                   ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
33471                                   ; (SYSINIT:212Bh)
33472                                   
33473                                               ;public DevEntry
33474                                   
33475 00002433 0000                     DevSize:    dw     0      ; size of the device driver being loaded(paras)
33476 00002435 0000                     DevLoadAddr:       dw     0       ; Mem addr where the device driver is 2 b loaded
33477 00002437 0000                     DevLoadEnd: dw     0      ; MaxAddr to which device can be loaded
33478 00002439 00000000                 DevEntry:   dd     0      ; Entry point to the device driver
33479 0000243D 00000000                 DevBrkAddr: dd     0      ; Break address of the device driver
33480                                   ; 30/12/2022
33481                                   ; 27/10/2022
33482 00002441 00                       ConvLoad:   db     0      ; Use conventional (dos 5 -style) InitDevLoad?
33483                                   ;
33484 00002442 00                       DevUMB:            db     0      ; byte indicating whether to load DDs in UMBs
33485 00002443 0000                     DevUMBAddr: dw     0      ; cuurent UMB used fro loading devices (paras)
33486 00002445 0000                     DevUMBSize: dw     0      ; Size of the current UMB being used   (paras)
33487 00002447 0000                     DevUMBFree: dw     0      ; Start of free are in the current UMB (paras)
33488                                   ;
33489 00002449 00000000                 DevXMSAddr: dd     0
33490                                   ;
33491 0000244D 0000                     DevExecAddr:       dw     0      ; Device load address parameter to Exec call
33492 0000244F 0000                     DevExecReloc:      dw     0      ; Device load relocation factor
33493                                   ;
33494 00002451 00                       DeviceHi:  db      0      ; Flag indicating whther the current device
33495                                                              ;  is being loaded into UMB
33496 00002452 0000                     DevSizeOption:     dw     0      ; SIZE= option
33497                                   ;
33498 00002454 00                       Int12Lied:  db     0      ; did we trap int 12 ?
33499 00002455 0000                     OldInt12Mem:       dw     0      ; value in 40:13h (int 12 ram)
33500 00002457 50524F544D414E24         ThreeComName:      db     'PROTMAN$'     ; 3Com Device name
33501                                   ;
33502 0000245F 00                       FirstUMBLinked:    db     0
33503 00002460 0000                     DevDOSData: dw     0      ; segment of DOS Data
33504 00002462 00000000                 DevCmdLine: dd     0      ; Current Command line
33505 00002466 00                       DevSavedDelim:     db     0      ; The delimiter which was replaced with null
33506                                                              ; to use the file name in the command line
33507                                   ; 13/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
33508                                   ;   ifdef  dblspace_hooks
33509 00002467 00                       MagicHomeFlag:     db     0      ; set non-zero when MagicDrv is final placed
33510                                   ;   endif
33511                                   
33512                                   ; ==========================================================================
33513                                   
33514                                   ; 31/03/2019 - Retro DOS v4.0
33515                                   
33516                                   ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
33517                                   ; (SYSINIT:215Eh)
33518                                   
33519                                   ;--------------------------------------------------------------------------
33520                                   ;
33521                                   ; procedure : doconf
33522                                   ;
33523                                   ;             Config file is parsed initially with this routine. For the
33524                                   ;             Subsequent passes 'multi_pass' entry is used .
33525                                   ;
33526                                   ;--------------------------------------------------------------------------
33527                                   
33528                                       ; 27/10/2022
33529                                   doconf:
33530 00002468 0E                           push    cs
33531 00002469 1F                           pop     ds
33532                                   
33533 0000246A B80037                       mov     ax,3700h
33534                                         ;mov       ax,(CHAR_OPER<<8)     ; get switch character
33535 0000246D CD21                         int     21h
33536 0000246F 8816[BC4B]                   mov     [command_line+1],dl   ; set in default command line
33537                                   
33538                                   ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21  IO.SYS)
33539                                   ; 27/10/2022
33540                                   ;;ifdef     MULTI_CONFIG
33541                                   ;   ;mov    [command_line-1],dl     ; save default switchchar
33542 00002473 8816[BA4B]                   mov     [def_swchr],dl ; 31/03/2019
33543                                   ;;endif     ;MULTI_CONFIG
33544                                   
33545 00002477 BA[D14A]                     mov     dx,config ;'\CONFIG.SYS' ;now pointing to file description
33546 0000247A B8003D                       mov     ax,3D00h
33547                                       ;mov    ax,OPEN<<8              ;open file "config.sys"
33548 0000247D F9                           stc                             ;in case of int 24
33549 0000247E CD21                         int     21h                     ;function request
33550 00002480 7309                         jnc     short noprob            ; brif opened okay
33551                                   
33552                                   ; 31/12/2022
33553                                   ; 27/10/2022
33554                                   ;;ifdef     MULTI_CONFIG
33555 00002482 E8A019                       call    kbd_read                ; we still want to give the guy
33556                                   ;                                   ; a chance to select clean boot!
33557                                   ;;endif                                      ; (ie, no autoexec.bat processing)
33558 00002485 C606[CD02]0B                 mov     byte [multi_pass_id],11     ; set it to unreasonable number
33559 0000248A C3                           retn
33560                                   noprob:                             ;get file size (note < 64k!!)
33561 0000248B 89C3                         mov     bx,ax  ; File handle
33562 0000248D 31C9                         xor     cx,cx                   ; 0
33563 0000248F 31D2                         xor     dx,dx                   ; 0
33564                                       ;mov    ax,4202h
33565 00002491 B80242                       mov     ax,(LSEEK<<8)|2
33566 00002494 CD21                         int     21h
33567 00002496 A3[5603]                     mov     [count],ax              ; dx:ax = file size ; 08/09/2023
33568                                                                       ; 08/09/2023 - Erdogan Tan - Note:
```

```
00002499 31D2                   xor     dx,dx                   ; dx already must be 0 here ; 08/09/2023
                                                                ; I am not removing 'xor dx,dx' here
                                                                ; for MSDOS compatibility.
                                                                ; ((Also PCDOS 7.1 has 'xor dx,dx' here))
                                                                ; (Error will be same if CONGIG.SYS file
                                                                ;  size > 64KB)
                                ;mov    ax,4200h
0000249B B80042                 mov     ax,LSEEK<<8             ;reset pointer to beginning of file
0000249E CD21                   int     21h

                                ; 31/12/2022 - Retro DOS v4.2
000024A0 8B16[A502]             mov     dx,[ALLOCLIM]           ;use current alloclim value
                                ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;mov    dx,[top_of_cdss]

000024A4 A1[5603]               mov     ax,[count]
000024A7 A3[D002]               mov     [config_size],ax        ;save the size of config.sys file.
000024AA E868EE                 call    ParaRound
000024AD 29C2                   sub     dx,ax

                              ; 31/12/2022
                              ; 27/10/2022
                              ;ifdef     MULTI_CONFIG
                              ;
                              ;  The size of the CONFIG.SYS workspace (for recreating the in-memory
                              ;  CONFIG.SYS image, and later for building the initial environment) need
                              ;  not be any larger than CONFIG.SYS itself, EXCEPT for the fact that
                              ;  we (may) add a variable to the environment that does not explicity appear
                              ;  in CONFIG.SYS, and that variable is CONFIG (as in CONFIG=COMMON).
                              ;  The default setting for CONFIG cannot result in more than 1 paragraph
                              ;  of extra space, so here we account for it (the worst case of course is
                              ;  when CONFIG.SYS is some very small size, like 0 -JTP)
                              ;
000024AF 4A                       dec     dx                      ;reserve 1 additional paragraph
000024B0 8916[6219]               mov     [config_wrkseg],dx      ;this is the segment to be used for
000024B4 29C2                     sub     dx,ax                   ;rebuilding the config.sys memory image
                              ;;endif    ;MULTI_CONFIG

000024B6 83EA11                 sub     dx,11h                  ;room for header

                                ; 31/12/2022
000024B9 8916[A502]             mov     [ALLOCLIM],dx           ;config starts here. new alloclim value.
000024BD 8916[A302]             mov     [CONFBOT],dx

                                ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;mov    [top_of_cdss],dx
                                ;call    TempCDS
                                ; 31/12/2022
                                ; 11/12/2022
                                ; ds <> cs
                                ;mov    dx,[cs:top_of_cdss]

                                ; 08/09/2023
                                ; ds = cs
000024C1 8B0E[5603]             mov     cx,[count]

000024C5 8EDA                   mov     ds,dx
000024C7 8EC2                   mov     es,dx

000024C9 31D2                   xor     dx,dx
                                ; 08/09/2023
                                ;mov    cx,[cs:count]
000024CB B43F                   mov     ah,3Fh
                                ;mov    ah,READ  ; 3Fh
000024CD F9                     stc                             ;in case of int 24
000024CE CD21                   int     21h                     ;function request
000024D0 9C                     pushf

                              ; find the eof mark in the file. if present,then trim length.

000024D1 50                     push    ax
000024D2 57                     push    di
000024D3 51                     push    cx
000024D4 B01A                   mov     al,1Ah                  ; eof mark
000024D6 89D7                   mov     di,dx                   ; point to buffer
000024D8 E305                   jcxz    puteol                  ; no chars
000024DA F2AE                   repnz   scasb                   ; find end
000024DC 7501                   jnz     short puteol            ; none found and count exhausted

                              ; we found a 1a. back up

000024DE 4F                     dec     di                      ; backup past 1Ah

                              ;  just for the halibut, stick in an extra eol

                              puteol:
000024DF B00D                   mov     al,cr ; 0Dh
000024E1 AA                     stosb
000024E2 B00A                   mov     al,lf  ;0Ah
000024E4 AA                     stosb
000024E5 29D7                   sub     di,dx                   ; difference moved
                                ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;mov    [cs:count],di           ; new count

                                ; 11/12/2022
                                ; 31/03/2019 - Retro DOS v4.0
000024E7 0E                     push    cs
000024E8 1F                     pop     ds

000024E9 893E[5603]             mov     [count],di              ; new count

000024ED 59                     pop     cx
000024EE 5F                     pop     di
000024EF 58                     pop     ax

                                ; 11/12/2022
                                ; 27/10/2022
                                ;push   cs
                                ;pop    ds

000024F0 50                     push    ax
                                ;mov    ah,CLOSE
000024F1 B43E                   mov     ah,3Eh
000024F3 CD21                   int     21h
000024F5 58                     pop     ax
000024F6 9D                     popf
000024F7 7204                   jc      short conferr           ;if not we've got a problem
000024F9 39C1                   cmp     cx,ax
000024FB 742C                   jz      short getcom            ;couldn't read the file
                              conferr:
000024FD BA[D14A]               mov     dx,config               ;want to print config error
00002500 E82525                 call    badfil
                                ; 14/04/2024
                              endconv:  ; 01/01/2023
```

```
33693 00002503 C3                          retn
33694
33695                                  ;--------------------------------------------------------------------------
33696                                  ;
33697                                  ; entry : multi_pass
33698                                  ;
33699                                  ;             called to execute device=,install= commands
33700                                  ;
33701                                  ;--------------------------------------------------------------------------
33702
33703                                      ; 27/10/2022
33704                                  multi_pass:
33705 00002504 0E                          push    cs
33706 00002505 1F                          pop     ds
33707
33708 00002506 803E[CD02]0A                cmp     byte [multi_pass_id],10
33709                                  ;jae_endconv:
33710 0000250B 73F6                        jae     short endconv         ; do nothing. just return.
33711
33712                                      ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
33713 0000250D FF36[A302]                  push    word [CONFBOT]
33714                                      ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
33715                                      ;push  word [top_of_cdss]
33716 00002511 07                          pop     es                    ; es -> confbot
33717
33718 00002512 8B36[5803]                  mov     si,[org_count]
33719 00002516 8936[5603]                  mov     [count],si            ; set count
33720 0000251A 31F6                        xor     si,si ; 0
33721 0000251C 8936[5A03]                      mov     [chrptr],si             ; reset chrptr
33722 00002520 8936[AF02]                      mov     [linecount],si          ; reset linecount
33723
33724 00002524 E88822                      call    getchr
33725 00002527 EB06                        jmp     short conflp
33726
33727                                      ; 14/04/2024
33728                                      ; 01/01/2023
33729                                  ;endconv:
33730                                      ;retn
33731
33732                                  getcom:
33733                                      ; 03/01/2023
33734                                      ; ds = cs
33735 00002529 E8C016                         call    organize              ; organize the file
33736 0000252C E88022                      call    getchr
33737                                  conflp:
33738 0000252F 72D2                        jc      short endconv
33739
33740 00002531 FF06[AF02]                      inc     word [linecount]  ; increase linecount
33741
33742                                      ; 08/09/2023
33743 00002535 30E4                        xor     ah,ah ; 0
33744                                      ;mov   byte [multdeviceflag],0      ; reset multdeviceflag.
33745                                      ;mov   byte [setdevmarkflag],0      ; reset setdevmarkflag.
33746 00002537 8826[6619]                  mov     [multdeviceflag],ah ; 0
33747 0000253B 8826[6919]                  mov     [setdevmarkflag],ah ; 0
33748
33749 0000253F 3C0A                        cmp     al,lf                 ; linefeed?
33750 00002541 7448                        je      short blank_line      ;  then ignore this line.
33751
33752                                  ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
33753                                  ; (SYSINIT:23CCh)
33754                                  ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
33755                                  ;%if 0
33756
33757                                  ;ifdef     MULTI_CONFIG
33758
33759                                  ;   If this is a genuine CONFIG.SYS command, then there should be a line
33760                                  ;   number immediately following it....
33761
33762 00002543 A2[6419]                      mov     [config_cmd],al         ; save original command code
33763                                      ;and   al,NOT CONFIG_OPTION_QUERY
33764 00002546 247F                        and     al,~CONFIG_OPTION_QUERY ; and al,7Fh
33765
33766                                      ; 08/09/2023
33767 00002548 3826[6519]                  cmp     [config_multi],ah ; 0
33768                                      ;cmp   byte [config_multi],0 ; is this a multi-config config.sys?
33769 0000254C 7427                            je      short not_final          ; no, line number is not embedded
33770
33771 0000254E 50                              push    ax                      ;
33772 0000254F E85D22                          call    getchr                  ; ignore end-of-image errors,
33773 00002552 88C4                            mov     ah,al                   ; because if there's an error
33774 00002554 E85822                          call    getchr                  ; fetching the line number that's
33775 00002557 86E0                            xchg    al,ah                   ; supposed to be there, the next
33776 00002559 A3[AF02]                        mov     [linecount],ax          ; getchr call will get the same error
33777 0000255C 58                              pop     ax
33778                                  ;
33779                                  ;   HACK: when 4DOS.COM is the shell and it doesn't have an environment from
33780                                  ;   which to obtain its original program name, it grovels through all of
33781                                  ;   memory to find the filename that was used to exec it; it wants to find
33782                                  ;   the SHELL= line in the in-memory copy of CONFIG.SYS, and it knows that
33783                                  ;   sysinit converts the SHELL= keyword to an 'S', so it expects to find an 'S'
33784                                  ;   immediately before the filename, but since we are now storing line # info
33785                                  ;   in the config.sys memory image, 4DOS fails to find the 'S' in the right
33786                                  ;   spot.
33787                                  ;
33788                                  ;   So, on the final pass of CONFIG.SYS, copy the command code (eg, 'S')
33789                                  ;   over the line number info, since we no longer need that info anyway. This
33790                                  ;   relies on the fact that getchr leaves ES:SI pointing to the last byte
33791                                  ;   retrieved.
33792                                  ;
33793 0000255D 803E[CD02]02                    cmp byte [multi_pass_id],2; final pass?
33794 00002562 7211                            jb  short not_final             ; no
33795                                          ;test      word [install_flag],have_install_cmd
33796 00002564 F606[CE02]01                test    byte [install_flag],have_install_cmd ; 1
33797 00002569 7407                            jz  short final           ; no install cmds, so yes it is
33798 0000256B 803E[CD02]03                    cmp byte [multi_pass_id],3; final pass?
33799 00002570 7203                            jb  short not_final             ; no
33800                                  final:                                  ;
33801 00002572 268804                      mov     [es:si],al            ; save backward-compatible command code
33802                                  not_final:                              ;
33803                                  ;endif
33804
33805                                  ; 31/12/2022
33806                                  ;%endif ; 27/10/2022
33807
33808 00002575 88C4                        mov     ah,al
33809 00002577 E83522                      call    getchr
33810 0000257A 7314                        jnc     short tryi
33811
33812 0000257C 803E[CD02]02                cmp     byte [multi_pass_id],2
33813                                      ;jae   short jae_endconv     ; do not show badop again for multi_pass.
33814                                      ; 27/10/2022
33815 00002581 7380                        jnb     short endconv
33816 00002583 E90009                      jmp     badop
```

```
                                coff:
                                    ; 11/12/2022
                                    ; ds = cs
                                    ;push  cs
                                    ;pop   ds
00002586 E81D22                     call   newline
00002589 EBA4                       jmp    short conflp   ; 13/05/2019

                                blank_line:
0000258B E82122                     call   getchr
0000258E EB9F                       jmp    short conflp

                                    ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ; 11/12/2022
                                ; (there is not a jump or call to here from anywhere!)
                                ;coff_p:
                                    ;push  cs
                                    ;pop   ds

                                ;to handle install= commands,we are going to use multi-pass.
                                ;the first pass handles the other commands and only set install_flag when
                                ;it finds any install command. the second pass will only handle the
                                ;install= command.

                                ;--------------------------------------------------------------------------
                                ;install command
                                ;--------------------------------------------------------------------------

                                    ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:2250h)
                                tryi:
00002590 803E[CD02]00               cmp    byte [multi_pass_id],0; the initial pass for DOS=HI
00002595 7503                       jne    short not_init_pass
00002597 E97F01                     jmp    multi_try_doshi
                                not_init_pass:
0000259A 803E[CD02]02               cmp    byte [multi_pass_id],2; the second pass was for ifs=
                                        ; 11/12/2022
                                    ;je    short multi_pass_coff2; now it is NOPs
0000259F 74E5                       je     short coff
                                    ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;je    short multi_pass_coff
                                                                   ; This pass can be made use of if
                                                                   ; we want do some config.sys process
                                                                   ; after device drivers are loaded
                                                                   ; and before install= commands
                                                                   ; are processed
000025A1 803E[CD02]03               cmp    byte [multi_pass_id],3; the third pass for install= ?
000025A6 741D                       je     short multi_try_i
000025A8 80FC48                         cmp     ah, CONFIG_DOS  ; 'H'
                                ; 11/12/2022
                                    ;je    short multi_pass_coff2
000025AB 74D9                       je     short coff
                                    ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;je    short multi_pass_coff

                                ;       make note of any INSTALL= or INSTALLHIGH= commands we find,
                                ;       but don't process them now.

000025AD 80FC49                         cmp     ah,CONFIG_INSTALL ; 'I'  ; install= command?
                                    ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
000025B0 7507                       jne    short precheck_installhigh ; the first pass is for normal operation.
                                    ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;jne   short tryb

                                    ;or    word [install_flag],have_install_cmd ; set the flag
000025B2 800E[CE02]01               or     byte [install_flag],have_install_cmd ; 1
                                multi_pass_coff2:
000025B7 EBCD                       jmp    short coff ; 13/05/2019      ; and handles the next command

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:2448h)
                                ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                precheck_installhigh:
000025B9 80FC57                     cmp     ah,CONFIG_INSTALLHIGH ; 'W' ; signifier for INSTALLHIGH
000025BC 756B                       jne     short tryb           ; carry on with normal processing
                                    ;or    word [install_flag],have_install_cmd
000025BE 800E[CE02]01               or     byte [install_flag],have_install_cmd ; 1
000025C3 EBC1                       jmp    short coff
                                ;%endif ; 27/10/2022

                                multi_try_i:
000025C5 80FC49                         cmp     ah,CONFIG_INSTALL ; 'I' ; install= command?
                                    ; 31/12/2022 - Retro DOS v4.2
000025C8 750A                       jne    short multi_try_n     ; no, check for installhigh
                                    ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;jne   short multi_pass_filter

                                ; 31/12/2022
                                ;%if 1
                                ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                ;ifdef      MULTI_CONFIG
000025CA E84F20                     call   query_user              ; query the user if config_cmd
000025CD 7241                       jc     short multi_pass_filter      ; has the CONFIG_OPTION_QUERY bit set
                                ;endif
                                ;%endif ; 27/10/2022

000025CF E8C7EC                     call   do_install_exec       ;install it.
000025D2 EBB2                       jmp    short coff            ;to handle next install= command.

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:2463h)
                                ;%if 1
                                ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0

                                multi_try_n:
000025D4 80FC57                         cmp     ah,CONFIG_INSTALLHIGH   ; installhigh= command?
000025D7 7537                           jne short multi_pass_filter      ; no. ignore this.
                                ;ifdef      MULTI_CONFIG
000025D9 E84020                         call    query_user              ; query the user if config_cmd
000025DC 7232                           jc      short multi_pass_filter  ; has the CONFIG_OPTION_QUERY bit set
                                ;endif

                                ;       The memory environment is in its normal DOS state, so do
                                ;       the standard calls to set the alloc strategy for loading high

000025DE B80058                     mov    ax,(ALLOCOPER<<8)|0 ; 5800h
000025E1 CD21                       int    21h                 ;get alloc strategy
000025E3 89C3                       mov    bx,ax
000025E5 53                             push    bx                      ; save for the return
```

```
33941
33942 000025E6 81CB8000                      or  bx,HIGH_FIRST  ; 80h  ;set alloc to HighFirst
33943 000025EA B80158                    mov    ax,(ALLOCOPER<<8)|1 ; 5801h
33944 000025ED CD21                      int    21h                 ;set alloc strategy
33945
33946 000025EF B80258                    mov     ax,(ALLOCOPER<<8)|2 ; 5802h
33947 000025F2 CD21                        int     21h                     ; get link state
33948 000025F4 30E4                        xor     ah,ah                   ; clear top byte
33949 000025F6 50                          push    ax                      ; save for return
33950
33951 000025F7 B80358                      mov ax,(ALLOCOPER<<8)|3 ; 5803h
33952 000025FA BB0100                    mov    bx,1
33953 000025FD CD21                      int    21h                 ;link in UMBs
33954
33955 000025FF E897EC                    call   do_install_exec     ;install it.
33956
33957 00002602 B80358                      mov     ax,(ALLOCOPER<<8)|3
33958 00002605 5B                          pop     bx                      ; recover original link state
33959 00002606 CD21                        int     21h
33960 00002608 5B                          pop     bx                      ; recover original alloc strategy
33961 00002609 B80158                      mov     ax,(ALLOCOPER<<8)|1
33962 0000260C CD21                        int     21h
33963
33964                                    ;jmp   short coff          ;to handle next install= command.
33965                                    ; 01/01/2023
33966 0000260E EBA7                      jmp    short multi_pass_coff2
33967
33968                                ;%endif ; 27/10/2022
33969
33970                                multi_pass_filter:
33971 00002610 80FC59                        cmp     ah,CONFIG_COMMENT ; 'Y' ; comment?
33972 00002613 740A                      je     short multi_pass_adjust
33973 00002615 80FC5A                        cmp     ah,CONFIG_UNKNOWN ; 'Z' ; bad command?
33974 00002618 7405                      je     short multi_pass_adjust
33975 0000261A 80FC30                        cmp     ah,CONFIG_REM  ; '0'     ; rem?
33976 0000261D 7508                      jne    short multi_pass_coff ; ignore the rest of the commands.
33977
33978                                multi_pass_adjust:             ; these commands need to
33979 0000261F FF0E[5A03]                dec    word [chrptr]       ;  adjust chrptr,count
33980 00002623 FF06[5603]                inc    word [count]        ;  for newline proc.
33981
33982                                multi_pass_coff:
33983                                    ; 11/12/2022
33984                                    ;jmp   short coff          ; to handle next install= commands.
33985                                    ; 01/01/2023
33986 00002627 EB8E                      jmp    short multi_pass_coff2
33987
33988                                ;--------------------------------------------------------------------------
33989                                ; buffer command
33990                                ;--------------------------------------------------------------------------
33991
33992                                ;****************************************************************************
33993                                ;                                                                           *
33994                                ; function: parse the parameters of buffers= command.                       *
33995                                ;                                                                           *
33996                                ; input :                                                                   *
33997                                ;   es:si -> parameters in command line.                                    *
33998                                ; output:                                                                   *
33999                                ;   buffers set                                                             *
34000                                ;   buffer_slash_x flag set if /x option chosen.                            *
34001                                ;   h_buffers set if secondary buffer cache specified.                      *
34002                                ;                                                                           *
34003                                ; subroutines to be called:                                                 *
34004                                ;   sysinit_parse                                                           *
34005                                ; logic:                                                                    *
34006                                ; {                                                                         *
34007                                ;   set di points to buf_parms;  /*parse control definition*/              *
34008                                ;   set dx,cx to 0;                                                         *
34009                                ;   reset buffer_slash_x;                                                   *
34010                                ;   while (end of command line)                                            *
34011                                ;   { sysinit_parse;                                                       *
34012                                ;     if (no error) then                                                    *
34013                                ;          if (result_val._$P_synonym_ptr == slash_e) then /*not a switch *
34014                                ;              buffer_slash_x = 1                                         *
34015                                ;          else if  (cx == 1) then           /* first positional */    *
34016                                ;                   buffers = result_val._$P_picked_val;                  *
34017                                ;               else  h_buffers = result_val._$P_picked_val;              *
34018                                ;     else {show error message;error exit}                                  *
34019                                ;   };                                                                      *
34020                                ;   if (buffer_slash_x is off & buffers > 99) then show_error;              *
34021                                ; };                                                                        *
34022                                ;                                                                           *
34023                                ;****************************************************************************
34024
34025                                    ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
34026                                    ; (SYSINIT:229Ch)
34027                                tryb:
34028 00002629 80FC42                       cmp     ah,CONFIG_BUFFERS ; 'B'
34029 0000262C 755C                      jne    short tryc
34030
34031                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
34032                                ; (SYSINIT:24BFh)
34033                                ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
34034                                ;%if 0
34035                                ;ifdef     MULTI_CONFIG
34036 0000262E E8EB1F                    call   query_user          ; query the user if config_cmd
34037 00002631 7257                      jc     short tryc          ; has the CONFIG_OPTION_QUERY bit set
34038                                ;endif
34039                                ;%endif ; 27/10/2022
34040
34041                                    ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
34042                                    ; 18/12/2022
34043 00002633 31C9                      xor    cx,cx
34044                                    ;mov   byte [p_buffer_slash_x],0 ; 31/03/2019
34045 00002635 880E[1522]                mov    [p_buffer_slash_x],cl ; 0
34046
34047 00002639 BF[CE21]                  mov    di,buf_parms
34048                                    ;xor   cx,cx   ; 18/12/2022
34049                                    ; 03/01/2023
34050                                    ;mov   dx,cx
34051                                do7:
34052 0000263C E82808                    call   sysinit_parse
34053 0000263F 7303                      jnc    short if7           ; parse error,
34054                                    ;call  badparm_p           ;  and show messages and end the search loop.
34055                                    ;;jmp  short sr7
34056                                    ; 31/12/2022
34057                                ;sr7:
34058                                    ;jmp   coff
34059                                    ; 03/01/2023
34060 00002641 E91207                    jmp    badparm_p_coff
34061                                if7:
34062 00002644 83F8FF                    cmp    ax,_$P_RC_EOL ; 0FFFFh; end of line?
34063 00002647 741A                      je     short en7           ;  then jmp to $endloop for semantic check
34064                                    ;cmp   word [result_val_swoff],switch_x
```

```
34065 00002649 813E[1922][0E22]           cmp     word [result_val+_$P_Result_Blk.SYNONYM_Ptr],switch_x
34066                                     ;jne    short if11
34067                                     ; 31/12/2022
34068 0000264F 74EB                       je      short do7 ;je short en11
34069
34070                                 ;   mov     byte [p_buffer_slash_x],1 ; set the flag M016
34071                                     ;jmp    short en11 ; 31/12/2022
34072                                 if11:
34073                                     ;mov    ax,[rv_dword]
34074 00002651 A1[1B22]                   mov     ax,[result_val+_$P_Result_Blk.Picked_Val]
34075 00002654 83F901                     cmp     cx,1
34076 00002657 7505                       jne     short if13
34077
34078 00002659 A3[1122]                   mov     [p_buffers],ax
34079                                     ;jmp    short en11
34080                                     ; 31/12/2022
34081 0000265C EBDE                       jmp     short do7
34082                                 if13:
34083 0000265E A3[1322]                   mov     [p_h_buffers],ax
34084                                 en11:
34085 00002661 EBD9                       jmp     short do7
34086                                 en7:
34087 00002663 833E[1122]63               cmp     word [p_buffers],99
34088 00002668 760B                       jbe     short if18
34089
34090                                 ;   cmp     byte [p_buffer_slash_x],0 ; M016
34091                                 ;   jne     short if18
34092
34093 0000266A E82508                     call    badparm_p
34094 0000266D C706[1322]0000             mov     word [p_h_buffers],0
34095 00002673 EB12                       jmp     short sr7
34096                                 if18:
34097 00002675 A1[1122]                   mov     ax,[p_buffers] ; we don't have any problem.
34098 00002678 A3[9902]                   mov     [buffers],ax   ; now,let's set it really.
34099
34100 0000267B A1[1322]                   mov     ax,[p_h_buffers]
34101 0000267E A3[9B02]                   mov     [h_buffers],ax
34102
34103                                 ;   mov     al,[p_buffer_slash_x] ; M016
34104                                 ;   mov     [buffer_slash_x],al
34105
34106 00002681 A1[AF02]                   mov     ax,[linecount]
34107 00002684 A3[B902]                   mov     [buffer_linenum],ax ; save the line number for the future use.
34108                                     ; 31/12/2022
34109                                     ;jmp    short sr7
34110                                     ; 03/01/2023
34111                                 sr7:
34112 00002687 E9FCFE                     jmp     coff
34113
34114                                 ;--------------------------------------------------------------------------
34115                                 ; break command
34116                                 ;--------------------------------------------------------------------------
34117
34118                                 ;**************************************************************************
34119                                 ;                                                                         *
34120                                 ; function: parse the parameters of break = command.                      *
34121                                 ;                                                                         *
34122                                 ; input :                                                                 *
34123                                 ;   es:si -> parameters in command line.                                  *
34124                                 ; output:                                                                 *
34125                                 ;   turn the control-c check on or off.                                   *
34126                                 ;                                                                         *
34127                                 ; subroutines to be called:                                               *
34128                                 ;   sysinit_parse                                                         *
34129                                 ; logic:                                                                  *
34130                                 ; {                                                                       *
34131                                 ;   set di to brk_parms;                                                  *
34132                                 ;   set dx,cx to 0;                                                       *
34133                                 ;   while (end of command line)                                           *
34134                                 ;   { sysinit_parse;                                                      *
34135                                 ;     if (no error) then                                                  *
34136                                 ;          if (result_val._$P_item_tag == 1) then        /*on        */ *
34137                                 ;             set p_ctrl_break,on;                                       *
34138                                 ;          else                                  /*off       */ *
34139                                 ;             set p_ctrl_break,off;                                      *
34140                                 ;     else {show message;error_exit};                                     *
34141                                 ;   };                                                                    *
34142                                 ;   if (no error) then                                                    *
34143                                 ;      dos function call to set ctrl_break check according to           *
34144                                 ; };                                                                      *
34145                                 ;                                                                         *
34146                                 ;**************************************************************************
34147
34148                                     ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
34149                                     ; (SYSINIT:22FFh)
34150                                 tryc:
34151 0000268A 80FC43                       cmp     ah,CONFIG_BREAK ; 'C'
34152 0000268D 7539                       jne     short trym
34153
34154                                 ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
34155                                 ; (SYSINIT:2527h)
34156                                 ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
34157                                 ;%if 0
34158                                 ;ifdef      MULTI_CONFIG
34159 0000268F E88A1F                     call    query_user              ; query the user if config_cmd
34160 00002692 7234                       jc      short trym              ; has the CONFIG_OPTION_QUERY bit set
34161                                 ;endif
34162                                 ;%endif ; 27/10/2022
34163
34164 00002694 BF[1F22]                   mov     di,brk_parms
34165 00002697 31C9                       xor     cx,cx
34166                                     ; 03/01/2023
34167                                     ;mov    dx,cx
34168                                 do22:
34169 00002699 E8CB07                     call    sysinit_parse
34170 0000269C 7303                       jnc     short if22              ; parse error
34171                                     ;call   badparm_p               ;  show message and end the search loop.
34172                                     ;;jmp   short sr22
34173                                     ; 31/12/2022
34174                                 ;sr22:
34175                                     ;jmp    coff
34176                                     ; 03/01/2023
34177 0000269E E9B506                     jmp     badparm_p_coff
34178                                 if22:
34179 000026A1 83F8FF                     cmp     ax,_$P_RC_EOL           ; end of line?
34180 000026A4 7415                       je      short en22              ; then end the $endloop
34181
34182                                     ;cmp    byte [result_val_itag],1
34183 000026A6 803E[1822]01               cmp     byte [result_val+_$P_Result_Blk.Item_Tag],1
34184 000026AB 7507                       jne     short if26
34185
34186 000026AD C606[4422]01               mov     byte [p_ctrl_break],1 ; turn it on
34187                                     ;jmp    short en26
34188                                     ; 31/12/2022
```

```
000026B2 EBE5                       jmp     short do22
                                if26:
000026B4 C606[4422]00               mov     byte [p_ctrl_break],0 ; turn it off
                                en26:
000026B9 EBDE                       jmp     short do22              ; we actually set the ctrl break
                                en22:
000026BB B433                       mov     ah,SET_CTRL_C_TRAPPING ; if we don't have any parse error.
000026BD B001                       mov     al,1
000026BF 8A16[4422]                 mov     dl,[p_ctrl_break]
000026C3 CD21                       int     21h
                                    ; 31/12/2022
                                    ;jmp    short sr22
                                    ; 03/01/2023
                                sr22:
000026C5 E9BEFE                     jmp     coff

                                ;--------------------------------------------------------------------------
                                ; multitrack command
                                ;--------------------------------------------------------------------------

                                ;****************************************************************************
                                ;                                                                           *
                                ; function: parse the parameters of multitrack= command.                   *
                                ;                                                                           *
                                ; input :                                                                  *
                                ;   es:si -> parameters in command line.                                    *
                                ; output:                                                                  *
                                ;   turn multrk_flag on or off.                                           *
                                ;                                                                           *
                                ; subroutines to be called:                                                *
                                ;   sysinit_parse                                                         *
                                ; logic:                                                                   *
                                ; {                                                                        *
                                ;   set di to brk_parms;                                                  *
                                ;   set dx,cx to 0;                                                       *
                                ;   while (end of command line)                                           *
                                ;   { sysinit_parse;                                                      *
                                ;     if (no error) then                                                  *
                                ;          if (result_val._$P_item_tag == 1) then        /*on         */   *
                                ;             set p_mtrk,on;                                            *
                                ;          else                                  /*off         */   *
                                ;             set p_mtrk,off;                                           *
                                ;     else {show message;error_exit};                                     *
                                ;   };                                                                    *
                                ;   if (no error) then                                                    *
                                ;      dos function call to set multrk_flag according to p_mtrk.          *
                                ;                                                                           *
                                ; };                                                                       *
                                ;                                                                           *
                                ;****************************************************************************

                                    ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                trym:
000026C8 80FC4D                        cmp     ah,CONFIG_MULTITRACK  ; 'M'
000026CB 7573                       jne     short tryu

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:2569h)
                                ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                ;ifdef      MULTI_CONFIG
000026CD E84C1F                     call    query_user      ; query the user if config_cmd
000026D0 726E                       jc      short tryu     ; has the CONFIG_OPTION_QUERY bit set
                                ;endif
                                ;%endif     ; 27/10/2022

000026D2 BF[2323]                   mov     di,mtrk_parms
000026D5 31C9                       xor     cx,cx
                                    ; 03/01/2023
                                    ;mov    dx,cx
                                do31:
000026D7 E88D07                     call    sysinit_parse
000026DA 7303                       jnc     short if31     ; parse error
                                    ;call   badparm_p      ;  show message and end the search loop.
                                    ;;jmp   short sr31
                                    ; 31/12/2022
                                ;sr31:
                                    ;jmp    coff
                                    ; 03/01/2023
000026DC E97706                     jmp     badparm_p_coff
                                if31:
000026DF 83F8FF                     cmp     ax,_$P_RC_EOL  ; end of line?
000026E2 7415                       je      short en31     ; then end the $endloop

                                    ;cmp    byte [result_val_itag],1
000026E4 803E[1822]01               cmp     byte [result_val+_$P_Result_Blk.Item_Tag],1
000026E9 7507                       jne     short if35

000026EB C606[3723]01               mov     byte [p_mtrk],1       ; turn it on temporarily.
                                    ;jmp    short en35
                                    ; 31/12/2022
000026F0 EBE5                       jmp     short do31
                                if35:
000026F2 C606[3723]00               mov     byte [p_mtrk],0       ; turn it off temporarily.
                                en35:
000026F7 EBDE                       jmp     short do31     ; we actually set the multrk_flag here
                                en31:
000026F9 1E                         push    ds
                                    ;;mov   ax,Bios_Data ; 70h
                                    ;mov    ax,KERNEL_SEGMENT ; 70h
                                    ; 21/10/2022
000026FA B87000                     mov     ax,DOSBIODATASEG ; 0070h
000026FD 8ED8                       mov     ds,ax

000026FF 2E803E[3723]00             cmp     byte [cs:p_mtrk],0
00002705 7508                       jne     short if39

00002707 C706[A004]0100             mov     word [multrk_flag],multrk_off2       ; 0001h
0000270D EB06                       jmp     short en39
                                if39:
0000270F C706[A004]8000             mov     word [multrk_flag],multrk_on  ; 0080h
                                en39:
00002715 1F                         pop     ds
                                    ; 31/12/2022
                                    ;jmp    short sr31
                                    ; 03/01/2023
                                sr31:
00002716 E96DFE                     jmp     coff

                                ;--------------------------------------------------------------------------
                                ; DOS=HIGH/LOW command
                                ;--------------------------------------------------------------------------

                                    ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
```

```
                                multi_try_doshi:
00002719 80FC48                         cmp     ah,CONFIG_DOS ; 'H'
0000271C 7403                       je      short it_is_h
                                skip_it:
0000271E E9EFFE                     jmp     multi_pass_filter
                                it_is_h:                            ; M003 - removed initing DevUMB
                                                                    ;        & runhigh
                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:25C1h)
                                ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                ;ifdef      MULTI_CONFIG
00002721 E8F81E                     call    query_user              ; query the user if config_cmd
00002724 72F8                       jc      short skip_it           ; has the CONFIG_OPTION_QUERY bit set
                                ;endif
                                ;%endif ; 27/10/2022

00002726 BF[9923]                   mov     di,dos_parms
00002729 31C9                       xor     cx,cx
                                    ; 03/01/2023
                                    ;mov    dx,cx
                                h_do_parse:
0000272B E83907                     call    sysinit_parse
0000272E 7303                       jnc     short h_parse_ok        ; parse error
                                h_badparm:
                                    ; 03/01/2023
                                    ;call   badparm_p               ; show message and end the search loop.
                                    ;;jmp   short h_end
                                    ; 11/12/2022
                                ;h_end:
                                    ;jmp    coff
                                    ; 03/01/2023
00002730 E92306                     jmp     badparm_p_coff
                                h_parse_ok:
00002733 83F8FF                     cmp     ax,_$P_RC_EOL           ; end of line?
00002736 7405                       je      short h_end             ; then end the $endloop
00002738 E8AE07                     call    ProcDOS
0000273B EBEE                       jmp     short h_do_parse
                                    ; 11/12/2022
                                    ; 03/01/2023
                                h_end:
0000273D E946FE                     jmp     coff

                                ;--------------------------------------------------------------------------------
                                ; devicehigh command
                                ;--------------------------------------------------------------------------------

                                    ; 28/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                tryu:
00002740 80FC55                         cmp     ah,CONFIG_DEVICEHIGH ; 'U'
00002743 7554                       jne     short tryd

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:25E9h)
                                ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                ;ifdef      MULTI_CONFIG
00002745 E8D41E                     call    query_user              ; query the user if config_cmd
00002748 724F                       jc      short tryd              ; has the CONFIG_OPTION_QUERY bit set
                                ;endif
                                ;%endif ; 28/10/2022

                                ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                    ; 01/01/2023
                                    ; ds = cs

0000274A E83108                     call    InitVar
0000274D E80510                     call    ParseSize               ; process the size= option
                                    ;jnc    short tryu_0
                                    ; 31/12/2022
00002750 720C                       jc      short tryu_1 ; 31/03/2019 - Retro DOS v4.0

                                ;%endif ; 28/10/2022

                                ; 31/12/2022
                                %if 0
                                    ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;mov    [cs:badparm_off], si  ; stash it there in case of an error
                                    ;mov    [cs:badparm_seg], es
                                    ; 11/12/2022
                                    ; ds = cs
                                    mov     [badparm_off], si
                                    mov     [badparm_seg], es

                                    ; 31/12/2022
                                    ;call   ParseSize
                                    ;jnc    short tryu_2   ; 28/10/2022

                                    ;call   badparm_p
                                    ;jmp    coff
                                    ; 03/01/2023
                                    jmp     badparm_p_coff
                                %endif

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:2606h)
                                ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                tryu_0:
                                    ;mov    ax,[cs:DevSizeOption]
                                    ; 31/12/2022
00002752 A1[5224]                   mov     ax,[DevSizeOption] ; ds = cs
00002755 09C0                       or      ax,ax
00002757 7510                       jnz     short tryu_2

00002759 E8B408                     call    ParseVar
0000275C 730B                       jnc     short tryu_2
                                tryu_1:
                                    ; 31/12/2022
                                    ; ds = cs
0000275E 8936[6B19]                 mov     [badparm_off], si
00002762 8C06[6D19]                 mov     [badparm_seg], es
                                    ;mov    [cs:badparm_off], si  ; If ParseVar up there failed, then
                                    ;mov    [cs:badparm_seg], es  ; ES:SI points to its problem area...

                                    ;call   badparm_p             ; so all we have to do is choke and
                                    ;jmp    coff                  ; die, rather verbosely.
                                    ; 03/01/2023
00002766 E9ED05                     jmp     badparm_p_coff

                                ;%endif ; 28/10/2022

                                tryu_2:
```

```
34437 00002769 56                       push    si
34438 0000276A 06                       push    es
34439
34440                                   ; 08/09/2023 - Retro DOS 4.2 IO.SYS (Optimization)
34441                                   ; MSDOS 6.21 IO.SYS - SYSINIT:2623h
34442                                   ; PCDOS 7.1 IBMBIO.COM - SYSINIT:2B17h
34443                                 tryu_3:
34444 0000276B 268A04                   mov     al,[es:si]
34445 0000276E 3C0D                     cmp     al,cr
34446                                   ; 14/04/2024
34447                                   ;je     short tryu_4
34448                                   ; 08/09/2023 - Retro DOS 5.0 (PCDOS 7.1 IBMBIO.COM)
34449 00002770 740C                     je      short tryu_5
34450 00002772 3C0A                     cmp     al,lf
34451 00002774 740A                     je      short tryu_4
34452 00002776 E81120                   call    delim
34453 00002779 7405                     jz      short tryu_4
34454 0000277B 46                       inc     si
34455 0000277C EBED                     jmp     short tryu_3
34456
34457                                   ; 14/04/2024
34458                                   ; 08/09/2023 - Retro DOS 5.0 (PCDOS 7.1 IBMBIO.COM)
34459                                 tryu_5:
34460 0000277E B020                     mov     al,20h ; ' '    ; blank instead of cr
34461
34462                                 tryu_4:
34463                                   ; 11/12/2022
34464                                   ; ds = cs
34465 00002780 A2[6624]                 mov     [DevSavedDelim],al
34466                                   ;mov    [cs:DevSavedDelim],al ; Save the delimiter before replacing
34467                                                                 ;  it with null
34468                                   ; 18/12/2022
34469 00002783 29DB                     sub     bx,bx
34470 00002785 26881C                   mov     [es:si],bl ; 0
34471                                   ;mov    byte [es:si],0
34472
34473 00002788 07                        pop     es
34474 00002789 5E                       pop     si      ; 14/04/2024
34475
34476                                 ;--------------------------------------------------------------------------
34477                                 ; BEGIN PATCH TO CHECK FOR NON-EXISTANT UMBs   -- t-richj 7-21-92
34478                                 ;--------------------------------------------------------------------------
34479
34480                                 ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
34481                                 ; (SYSINIT:2642h)
34482                                 ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
34483                                 ;%if 0
34484                                 ; 10/07/2023 - Retro DOS v4.2 IO.SYS (optimized)
34485                                 ; MSDOS 6.21 IO.SYS - SYSINIT:2642h
34486                                 %if 1
34487                                   ; 01/01/2023
34488                                   ; ds = cs
34489 0000278A E8F00C                   call    UmbTest                 ; See if UMBs are around...
34490                                   ; 01/01/2023
34491                                   ;jnc    short NrmTst            ; ...yep. So do that normal thang.
34492
34493                                   ;mov    byte [cs:DeviceHi],0    ; ...nope... so load low.
34494                                   ; 31/12/2022
34495                                   ; ds = cs, bx = 0
34496                                   ;mov    byte [DeviceHi],bl ; 0
34497                                   ;jmp    short LoadDevice
34498                                   ; 01/01/2023
34499 0000278D 7222                     jc      short LoadDevice ; bl = 0
34500                                 %endif
34501                                 ;%endif
34502                                 ;--------------------------------------------------------------------------
34503                                 ; END PATCH TO CHECK FOR NON-EXISTANT UMBs   -- t-richj 7-21-92
34504                                 ;--------------------------------------------------------------------------
34505
34506                                 NrmTst:
34507                                   ; 11/12/2022
34508                                   ; ds = cs
34509                                   ;;mov   byte [cs:DeviceHi],0
34510                                   ;mov    byte [DeviceHi],0
34511                                   ; 18/12/2022
34512                                   ; bx = 0
34513 0000278F 381E[4224]               cmp     [DevUMB],bl ; 0
34514                                   ;cmp    byte [DevUMB],0
34515                                   ;;cmp   byte [cs:DevUMB],0      ; do we support UMBs
34516 00002793 741C                     je      short LoadDevice        ; no, we don't
34517                                   ;mov    byte [cs:DeviceHi],1
34518                                   ; 11/12/2022
34519                                   ;mov    byte [DeviceHi],1
34520                                   ; 18/12/2022
34521 00002795 FEC3                     inc     bl ; mov bl,1 ; (*)
34522                                   ; 11/12/2022
34523                                   ;jmp    short LoadDevice2       ; 11/12/2022
34524 00002797 EB18                     jmp     short LoadDevice
34525
34526                                 ;--------------------------------------------------------------------------
34527                                 ; device command
34528                                 ;--------------------------------------------------------------------------
34529
34530                                   ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
34531                                   ; (SYSINIT:2665h)
34532
34533                                   ; 28/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
34534                                   ; (SYSINIT:2401h)
34535                                 tryd:
34536                                   ; 11/12/2022
34537                                   ;xor    bx,bx ; 31/12/2022
34538                                   ;
34539 00002799 80FC44                     cmp     ah,CONFIG_DEVICE ; 'D'
34540 0000279C 7403                     je      short gotd
34541                                 skip_it2:
34542 0000279E E9FC02                   jmp     tryq
34543                                 gotd:
34544
34545                                 ; 31/12/2022 - Retro DOS v4.2
34546                                 ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
34547                                 ;%if 0
34548                                 ;ifdef      MULTI_CONFIG
34549 000027A1 E8781E                   call    query_user              ; query the user if config_cmd
34550 000027A4 72F8                     jc      short skip_it2          ; has the CONFIG_OPTION_QUERY bit set
34551                                 ;endif
34552                                 ;%endif ; 28/10/2022
34553
34554                                   ; 31/12/2022
34555 000027A6 29DB                     sub     bx,bx
34556                                   ; bx = 0
34557                                   ; 11/12/2022
34558                                   ; ds = cs
34559                                   ;mov    byte [DeviceHi],0
34560                                   ;mov    word [DevSizeOption],0
```

```
34561 000027A8 891E[5224]                mov     [DevSizeOption],bx ; 0
34562 000027AC C606[6624]20              mov     byte [DevSavedDelim],' '
34563                                    ;mov    byte [cs:DeviceHi],0  ; not to be loaded in UMB ;M007
34564                                    ;mov    word [cs:DevSizeOption],0
34565                                    ;mov    byte [cs:DevSavedDelim],' ' ; In case of DEVICE= the null has to
34566                                                                        ;  be replaced with a ' '
34567                                LoadDevice:                             ; device= or devicehigh= command.
34568                                    ; 11/12/2022
34569                                    ;mov    byte [DeviceHi],0
34570 000027B1 881E[5124]                mov     byte [DeviceHi],bl     ; 0 or 1 (*)
34571                                LoadDevice2:
34572                                    ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
34573                                    ;
34574                                    ;push    cs
34575                                       ;pop     ds
34576                                    ;
34577                                    ;mov    [bpb_addr],si          ; pass the command line to the dvice
34578                                    ;mov    [bpb_addr+2],es
34579                                    ;
34580                                    ;mov    [DevCmdLine],si               ; save it for ourself
34581                                    ;mov    [DevCmdLine+2],es
34582                                    ;
34583                                    ;mov    byte [driver_units],0 ; clear total block units for driver
34584
34585                                    ; 11/12/2022
34586                                    ; ds = cs
34587                                    ;mov    bx,cs
34588                                    ;mov    ds,bx
34589
34590                                    ;mov    [cs:bpb_addr],si       ; pass the command line to the dvice
34591 000027B5 8936[8103]                mov     [bpb_addr],si
34592                                    ;mov    [cs:bpb_addr+2],es
34593 000027B9 8C06[8303]                mov     [bpb_addr+2],es
34594
34595                                    ;mov    [cs:DevCmdLine],si     ; save it for ourself
34596 000027BD 8936[6224]                mov     [DevCmdLine],si
34597                                    ;mov    [cs:DevCmdLine+2],es
34598 000027C1 8C06[6424]                mov     [DevCmdLine+2],es
34599
34600                                    ; 31/12/2022 - Retro DOS v4.2
34601 000027C5 C606[6A19]00              mov     byte [driver_units],0 ; clear total block units for driver
34602
34603 000027CA E82520                    call    round
34604
34605 000027CD E8910E                    call    SizeDevice
34606 000027D0 723F                      jc      short BadFile
34607
34608                                    ; 11/12/2022
34609                                    ; ds = cs
34610
34611                                ; - Begin DeviceHigh primary logic changes ---------------------------------
34612
34613                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
34614                                ; (SYSINIT:26A4h)
34615
34616                                ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
34617                                ;%if 0
34618 000027D2 C606[4124]01              mov     byte [ConvLoad],1      ; Doesn't matter if DeviceHi==0
34619
34620                                    ; 22/07/2023
34621                                    ;mov    al,[DeviceHi]          ; If not using upper memory,
34622 000027D7 800E[5124]00              or      byte [DeviceHi],0      ; Skip all this and go on to
34623                                    ; 10/07/2023
34624                                    ;or     al,al
34625 000027DC 741E                      jz      short DevConvLoad      ; the actual load.
34626
34627                                    ;call   GetLoadUMB             ; Returns first UMB spec'ed in AX
34628 000027DE A0[FF23]                  mov     al,[UmbLoad]   ; 19/04/2019 - Retro DOS v4.0
34629
34630 000027E1 3CFF                      cmp     al,-1                  ; If umb0 not specified, it's old style
34631 000027E3 7417                      jz      short DevConvLoad      ; so load high even if SIZE= is smaller
34632
34633 000027E5 FE0E[4124]                dec     byte [ConvLoad] ; 0    ; They specified /L, so use new loader
34634
34635 000027E9 E8590A                    call    GetLoadSize            ; Returns size of first UMB specified
34636 000027EC 09C0                      or      ax,ax
34637 000027EE 7406                      jz      short tryd_1           ; If size1 not specified, nada to do:
34638
34639 000027F0 3B06[3324]                cmp     ax,[DevSize]           ; /L:...,Size < DevSize?
34640 000027F4 7D06                      jge     short DevConvLoad
34641                                tryd_1:
34642 000027F6 A1[3324]                  mov     ax,[DevSize]           ; Size < DevSize, so write DevSize as
34643 000027F9 E8550A                    call    StoLoadSize            ; minsize for load UMB.
34644
34645                                ;%endif ; 28/10/2022
34646
34647                                ; - End DeviceHigh primary logic changes -----------------------------------
34648
34649                                DevConvLoad:
34650                                    ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
34651 000027FC E8660D                    call    InitDevLoad
34652
34653                                    ; 11/12/2022
34654                                    ; ds = cs
34655 000027FF A1[3524]                  mov     ax,[DevLoadAddr]
34656 00002802 0306[3324]                add     ax,[DevSize]
34657 00002806 7206                      jc      short NoMem
34658 00002808 3906[3724]                cmp     [DevLoadEnd],ax
34659 0000280C 7315                      jae     short LoadDev
34660
34661                                    ; 11/12/2022
34662                                    ;mov    ax,[cs:DevLoadAddr]
34663                                    ;add    ax,[cs:DevSize]
34664                                    ;jc     short NoMem
34665                                    ;cmp    [cs:DevLoadEnd],ax
34666                                    ;jae    short LoadDev
34667                                NoMem:
34668                                    ; 11/12/2022
34669                                    ; ds = cs
34670                                    ;jmp    mem_err
34671 0000280E E92020                    jmp     mem_err2
34672
34673                                BadFile:
34674                                    ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
34675                                    ;;call  RetFromUM              ; Does nothing if didn't call HideUMBs
34676                                    ;;cmp   byte [es:si],' '
34677                                       ;;jae      short tryd_2
34678                                    ; 31/12/2022
34679                                    ;cmp    byte [es:si],0Dh       ; cr
34680                                       ;jne       short tryd_2
34681                                    ;jmp    badop
34682                                    ; 31/12/2022
34683                                    ; ds = cs
34684                                    ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
```

```
                                    ; (SYSINIT:26E6h)
00002811 E8AE0E                     call    RetFromUM               ; Does nothing if didn't call HideUMBs
00002814 26803C20                   cmp     byte [es:si],' '
                                    ;cmp    byte [es:si],20h ; space
00002818 7303                       jnb     short tryd_2
0000281A E96906                     jmp     badop
                                tryd_2:
0000281D E80C22                     call    badload
00002820 E963FD                     jmp     coff

                                LoadDev:
00002823 06                         push    es
00002824 1F                         pop     ds

00002825 89F2                       mov     dx,si                   ;ds:dx points to file name
00002827 E87C0E                     call    ExecDev                 ; load device driver using exec call
                                badldreset:
0000282A 1E                         push    ds
0000282B 07                         pop     es                      ;es:si back to config.sys
0000282C 0E                         push    cs
0000282D 1F                         pop     ds                      ;ds back to sysinit
0000282E 72E1                       jc      short BadFile
                                goodld:
                                    ; 11/12/2022
                                    ; ds = cs

00002830 06                         push    es ; + ; 31/12/2022
00002831 56                         push    si ; ++
00002832 E89E0E                     call    RemoveNull
00002835 06                         push    es
00002836 56                         push    si

00002837 0E                         push    cs
00002838 07                         pop     es

00002839 1E                         push    ds ; **  ; ds = cs
0000283A 56                         push    si

                                    ;lds    si,[cs:DevEntry]        ; peeks the header attribute
                                    ; 31/12/2022
                                    ; ds = cs
0000283B C536[3924]                 lds     si,[DevEntry]

                                    ;test   word [si+4],8000h
                                    ; 11/12/2022
0000283F F6440580                   test    byte [si+SYSDEV.ATT+1],DEVTYP>>8
                                    ;test   word [si+SYSDEV.ATT],DEVTYP ; block device driver?
00002843 7514                       jnz     short got_device_com_cont   ; no.

00002845 2EC536[6D02]               lds     si,[cs:DOSINFO]             ; ds:si -> sys_var
                                    ;cmp    byte [si+32],26
0000284A 807C201A                   cmp     byte [si+SYSI_NUMIO],26     ; no more than 26 drive number
0000284E 7209                       jb      short got_device_com_cont

00002850 5E                         pop     si
00002851 1F                         pop     ds ; **

00002852 5E                         pop     si                      ; clear the stack
00002853 07                         pop     es

                                    ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;call   RetFromUM
                                    ; 31/12/2022
                                    ; ds = cs ; **
00002854 E86B0E                     call    RetFromUM               ; Do this before we leave

                                    ;jmp    short badnumblock
                                    ; 31/12/2022
00002857 EB73                       jmp     short badnumblock2  ; ds = cs

                                got_device_com_cont:
00002859 5E                         pop     si
0000285A 1F                         pop     ds

                                    ; 11/12/2022
                                    ; ds = cs

0000285B E8AE06                     call    LieInt12Mem
0000285E E80B07                     call    UpdatePDB               ; update the PSP:2 value M020

                                    ; 11/12/2022
                                    ; ds = cs
                                    ; 08/09/2023
00002861 31C0                       xor     ax, ax ; 0
00002863 3806[6619]                 cmp     byte [multdeviceflag],al ; 0
                                    ;cmp    byte [multdeviceflag],0
                                    ;cmp    byte [cs:multdeviceflag],0 ; Pass limit only for the 1st device
                                                                       ;  driver in the file ; M027
00002867 750B                       jne     short skip_pass_limit ;                     ; M027

                                    ; 11/12/2022
                                    ; ds = cs
                                    ;mov    word [cs:break_addr],0; pass the limit to the DD
                                    ;mov    bx,[cs:DevLoadEnd]
                                    ;mov    [cs:break_addr+2],bx

                                    ;mov    word [break_addr],0
                                    ; 08/09/2023
00002869 A3[7D03]                   mov     [break_addr],ax ; 0
0000286C 8B1E[3724]                 mov     bx,[DevLoadEnd]
00002870 891E[7F03]                 mov     [break_addr+2],bx

                                skip_pass_limit:
                                ;   Note: sysi_numio (in DOS DATA) currently reflects the REAL
                                ;   number of installed devices (including DblSpace drives) where
                                ;   "drivenumber" is the number that the next block device will
                                ;   be assigned to. Because some naughty device drivers (like
                                ;   interlnk) look at the internal DOS variable instead of the
                                ;   value we pass it, we'll temporarily stick our value into
                                ;   DOS DATA while we're initializing the device drivers.
                                ;
                                ;   Note that this will make it impossible for this device
                                ;   driver to access the DblSpace drive letters, whether
                                ;   they are swapped-hosts or unswapped compressed drives,
                                ;   during its initialization phase.

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:2752h)
                                ; 29/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                    ; 31/12/2022
                                    ;push   ds

                                    ;lds    bx,[cs:DOSINFO]             ; ds:bx -> sys_var
```

```
                                    ; 31/12/2022
                                    ; ds = cs
                                    ; 08/09/2023
                                    ;lds    bx,[DOSINFO]          ; ds:bx -> sys_var

                                    ;mov    al,[cs:drivenumber]   ; temporarily use this next drv value
                                    ;mov    [cs:devdrivenum],al   ; pass drive number in packet to driver
                                    ;mov    ah,al

                                    ; 08/09/2023
                                    ; ds = cs
00002874 A0[8503]                   mov     al,[drivenumber]      ; temporarily use this next drv value
00002877 A2[8503]                   mov     [devdrivenum],al      ; pass drive number in packet to driver
0000287A 88C4                       mov     ah,al
0000287C C51E[6D02]                 lds     bx,[DOSINFO]          ; ds:bx -> sys_var

00002880 874720                     xchg    ax,[bx+SYSI_NUMIO]    ; swap with existing values
                                    ; 31/12/2022
                                    ;pop    ds

00002883 50                         push    ax                    ; save real sysi_numio/ncds in ax

                                  ;%endif ; 29/10/2022

                                    ; 29/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:24B9h)

00002884 BB0600                     mov     bx,SYSDEV.STRAT ; 6
00002887 E8B01F                     call    calldev               ; calldev (sdevstrat);
0000288A BB0800                     mov     bx,SYSDEV.INT ; 8
0000288D E8AA1F                     call    calldev               ; calldev (sdevint);

                                    ; 11/12/2022
                                    ; ds <> cs (from calldev) ; 31/12/2022

                                  ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                  ; (SYSINIT:2773h)
                                  ; 29/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                  ;%if 0
00002890 58                         pop     ax                    ; get real sysi_numio value
                                    ; 31/12/2022
                                    ;push   ds
00002891 2EC51E[6D02]               lds     bx,[cs:DOSINFO]             ; ds:bx -> sys_var
00002896 894720                     mov     [bx+SYSI_NUMIO],ax    ; swap with existing values
                                    ; 31/12/2022
                                    ;pop    ds

                                  ;%endif ; 29/10/2022

                                    ; 11/12/2022
00002899 0E                         push    cs
0000289A 1F                         pop     ds

0000289B E89C06                     call    TrueInt12Mem

                                    ; 11/12/2022
                                    ; ds = cs
                                    ;mov    ax,[cs:break_addr]    ; move break addr from the req packet
                                    ;mov    [cs:DevBrkAddr],ax
                                    ;mov    ax,[cs:break_addr+2]
                                    ;mov    [cs:DevBrkAddr+2],ax
0000289E A1[7D03]                   mov     ax,[break_addr]
000028A1 A3[3D24]                   mov     [DevBrkAddr],ax
000028A4 A1[7F03]                   mov     ax,[break_addr+2]
000028A7 A3[3F24]                   mov     [DevBrkAddr+2],ax

                                    ; 29/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;call   RetFromUM             ; There we go... all done.
                                    ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:2791h)
000028AA E8150E                     call    RetFromUM             ; There we go... all done.

                                    ; 31/12/2022
                                    ; ds = cs

                                    ; 11/12/2022
000028AD 803E[4224]00               cmp     byte [DevUMB],0
                                    ;cmp    byte [cs:DevUMB],0
000028B2 7403                       je      short tryd_3
000028B4 E80010                     call    AllocUMB
                                    ; 31/12/2022
                                    ; ds = cs
                                  tryd_3:

                                  ;ifndef ROMDOS
                                  ;------ If we are waiting to be moved into hma lets try it now !!!

                                    ; 11/12/2022
                                    ; ds = cs

                                    ;cmp    byte [cs:runhigh],0FFh
000028B7 803E[6C02]FF               cmp     byte [runhigh],0FFh ; 11/12/2022
000028BC 7503                       jne     short tryd_4

                                    ; 11/12/2022
                                    ; ds = cs
000028BE E8D7E1                     call    TryToMovDOSHi         ; move DOS into HMA if reqd
                                  tryd_4:
                                  ;endif ; ROMDOS

000028C1 5E                         pop     si
000028C2 1F                         pop     ds
000028C3 C60400                     mov     byte [si],0           ; *p = 0;

000028C6 0E                         push    cs
000028C7 1F                         pop     ds

000028C8 EB1F                       jmp     short was_device_com

                                  ;--------------------------------------------------------------------------

                                  ; 02/04/2019 - Retro DOS v4.0

                                  ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                  ; (SYSINIT:27B3h)

                                  badnumblock:
000028CA 0E                         push    cs
000028CB 1F                         pop     ds
                                  badnumblock2:       ; 31/12/2022 (ds=cs)
000028CC BA[7051]                   mov     dx,badblock
000028CF E88221                     call    print

                                  ;------ fall thru ---------------------------------------------------------
```

```
34933
34934                                   ; 31/12/2022 - Retro DOS v4.2
34935
34936                                 erase_dev_do:                           ; modified to show message "error in config.sys..."
34937
34938                                   ;call  CheckDoubleSpace ; MSDOS 6.21 IO.SYS SYSINIT:27BBh
34939                                                           ; (Note: 'call CheckDoubleSpace'
34940                                                           ; has been removed at 'erase_dev_do:' pos
34941                                                           ; in PCDOS 7.1 IBMBIO.COM - SYSINIT:2CBAh)
34942                                                           ; Erdogan Tan - 10/07/2023
34943 000028D2 5E                       pop     si ; ++
34944 000028D3 07                       pop     es ; + ; 31/12/2022
34945
34946 000028D4 0E                       push    cs
34947 000028D5 1F                       pop     ds
34948
34949                                 skip1_resetmemhi:
34950                                   ; 11/12/2022
34951                                   ; ds = cs
34952 000028D6 833E[8603]00             cmp     word [configmsgflag],0
34953                                   ;cmp    word [cs:configmsgflag],0
34954 000028DB 7409                     je      short no_error_line_msg
34955
34956 000028DD E8DA05                   call    error_line              ; no "error in config.sys" msg for device driver. dcr d493
34957                                   ; 11/12/2022
34958                                   ; ds = cs
34959                                   ;mov    word [cs:configmsgflag],0
34960 000028E0 C706[8603]0000           mov     word [configmsgflag],0; set the default value again.
34961
34962                                 no_error_line_msg:
34963 000028E6 E99DFC                   jmp     coff
34964
34965                                 ;--------------------------------------------------------------------------
34966
34967                                 was_device_com:
34968                                   ; 14/12/2022
34969                                   ; ds = cs
34970 000028E9 A1[3F24]                 mov     ax,[DevBrkAddr+2]
34971                                   ;mov    ax,[cs:DevBrkAddr+2] ; 13/05/2019
34972 000028EC 3B06[3724]               cmp     ax,[DevLoadEnd]
34973                                   ;cmp    ax,[cs:DevLoadEnd]
34974 000028F0 7605                     jbe     short breakok
34975
34976 000028F2 5E                       pop     si
34977 000028F3 07                       pop     es
34978 000028F4 E91AFF                   jmp     BadFile
34979
34980                                 breakok:
34981                                   ; 14/12/2022
34982                                   ; ds = cs
34983 000028F7 C43E[6D02]               les     di,[DOSINFO]
34984 000028FB C516[3924]               lds     dx,[DevEntry]
34985                                   ;lds    dx,[cs:DevEntry]        ;set ds:dx to header
34986 000028FF 89D6                     mov     si,dx
34987
34988                                   ; 14/11/2022
34989                                   ;les    di,[cs:DOSINFO]         ;es:di point to dos info
34990
34991                                   ; 14/12/2022
34992                                   ; ds <> cs
34993
34994                                   ;mov    ax,[si+4]
34995 00002901 8B4404                   mov     ax,[si+SYSDEV.ATT]      ;get attributes
34996                                   ; 12/12/2022
34997 00002904 F6C480                   test    ah,DEVTYP>>8 ; 80h
34998                                   ;test   ax,DEVTYP ; 8000h       ;test if block dev
34999 00002907 7426                     jz      short isblock
35000
35001                                 ;------ lets deal with character devices
35002
35003 00002909 2E800E[6919]02           or      byte [cs:setdevmarkflag],for_devmark ; 2
35004 0000290F E8F40D                   call    DevSetBreak             ;go ahead and alloc mem for device
35005                                 jc_edd:
35006 00002912 72BE                     jc      short erase_dev_do      ;device driver's init routine failed.
35007
35008                                   ; 12/12/2022
35009 00002914 A801                     test    al,ISCIN
35010                                   ;test   ax,ISCIN ; 1            ;is it a console in?
35011 00002916 7408                     jz      short tryclk
35012
35013 00002918 2689550C                 mov     [es:di+SYSI_CON],dx   ; es:di+12
35014 0000291C 268C5D0E                 mov     [es:di+SYSI_CON+2],ds ; es:di+14
35015                                 tryclk:
35016                                   ; 12/12/2022
35017 00002920 A808                     test    al,ISCLOCK
35018                                   ;test   ax,ISCLOCK ; 8          ;is it a clock device?
35019 00002922 7408                     jz      short golink
35020
35021 00002924 26895508                 mov     [es:di+SYSI_CLOCK],dx ; es:di+8
35022 00002928 268C5D0A                 mov     [es:di+SYSI_CLOCK+2],ds ; es:di+10
35023                                 golink:
35024 0000292C E9DF00                   jmp     linkit
35025
35026                                 ;------ deal with block device drivers
35027
35028                                 isblock:
35029 0000292F 2EA0[7C03]               mov     al,[cs:unitcount]       ;if no units found,erase the device
35030 00002933 08C0                     or      al,al
35031 00002935 749B                     jz      short erase_dev_do
35032                                   ;mov    [si+10],al
35033 00002937 88440A                   mov     [si+SYSDEV.NAME],al   ; number of units in name field
35034                                   ; 29/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
35035                                   ;add    [cs:driver_units],al
35036                                   ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
35037 0000293A 2E0006[6A19]             add     [cs:driver_units],al  ; keep total for all drivers in file
35038                                 perdrv:
35039 0000293F 98                       cbw                             ; warning no device > 127 units
35040 00002940 89C1                     mov     cx,ax
35041 00002942 88E6                     mov     dh,ah
35042                                   ;mov    dl,[es:di+32]
35043 00002944 268A5520                 mov     dl,[es:di+SYSI_NUMIO] ;get number of devices
35044 00002948 88D4                     mov     ah,dl
35045 0000294A 00C4                     add     ah,al                   ; check for too many devices
35046 0000294C 80FC1A                   cmp     ah,26                   ; 'A' - 'Z' is 26 devices
35047 0000294F 7603                     jbe     short ok_block
35048 00002951 E976FF                   jmp     badnumblock
35049
35050                                 ok_block:
35051 00002954 2E800E[6919]02           or      byte [cs:setdevmarkflag],for_devmark ; 2
35052 0000295A E8A90D                   call    DevSetBreak             ; alloc the device
35053 0000295D 72B3                     jc      short jc_edd
35054 0000295F 26004520                 add     [es:di+SYSI_NUMIO],al ; update the amount
35055
35056 00002963 2E0006[8503]             add     [cs:drivenumber],al   ; remember amount for next device
```

```
35057 00002968 2EC51E[8103]              lds    bx,[cs:bpb_addr]      ; point to bpb array
35058                                  perunit:
35059 0000296D 2EC42E[6D02]              les    bp,[cs:DOSINFO]
35060                                    ;les   bp,[es:bp+SYSI_DPB]   ; get first dpb
35061                                    ; 11/12/2022
35062 00002972 26C46E00                  les    bp,[es:bp]
35063                                    ; 29/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
35064                                    ;les   bp,[es:bp+0]          ; [es:bp+SYSI_DPB]
35065                                  scandpb:
35066                                    ;cmp   word [es:bp+25],-1
35067 00002976 26837E19FF                cmp    word [es:bp+DPB.NEXT_DPB],-1
35068 0000297B 7406                      je     short foundpb
35069                                    ;les   bp,[es:bp+25]
35070 0000297D 26C46E19                  les    bp,[es:bp+DPB.NEXT_DPB]
35071 00002981 EBF3                      jmp    short scandpb
35072                                  foundpb:
35073 00002983 2EA1[3D24]                mov    ax,[cs:DevBrkAddr]
35074 00002987 26894619                  mov    [es:bp+DPB.NEXT_DPB],ax
35075 0000298B 2EA1[3F24]                mov    ax,[cs:DevBrkAddr+2]
35076 0000298F 2689461B                  mov    [es:bp+DPB.NEXT_DPB+2],ax
35077
35078 00002993 2EC42E[3D24]              les    bp,[cs:DevBrkAddr]
35079 00002998 2E8306[3D24]3D            add    word [cs:DevBrkAddr],DPBSIZ ; 33
35080                                                                ; 08/09/2023
35081                                                                ; (61 in PCDOS 7.1 IBMBIO.COM)
35082 0000299E E8440D                    call   RoundBreakAddr
35083
35084 000029A1 26C74619FFFF              mov    word [es:bp+DPB.NEXT_DPB],-1
35085 000029A7 26C64618FF                mov    byte [es:bp+DPB.FIRST_ACCESS],-1
35086
35087 000029AC 8B37                      mov    si,[bx]               ;ds:si points to bpb
35088 000029AE 43                        inc    bx
35089 000029AF 43                        inc    bx                    ;point to next guy
35090                                    ;mov   [es:bp+DPB.DRIVE],dx
35091                                    ; 11/12/2022
35092 000029B0 26895600                  mov    [es:bp],dx ; 13/05/2019
35093                                    ; 29/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
35094                                    ;mov   [es:bp+0],dx          ; [es:bp+DPB.DRIVE]
35095
35096                                    ; 13/04/2024 - Retro DOS v5.0
35097                                    ; PCDOS 7.1 IBMBIO.COM
35098                                    ;;;
35099 000029B4 52                        push   dx
35100 000029B5 51                        push   cx             ; initialize FAT32 extended DPB parameters/fields
35101 000029B6 BA5241                    mov    dx,4152h       ; 'AR' signature for FAT32 extended DPB
35102 000029B9 31C9                      xor    cx,cx  ; 0
35103                                    ;mov   [es:bp+1Dh],cx
35104 000029BB 26894E1D                  mov    [es:bp+DPB.NEXT_FREE],cx ; last allocated cluster #
35105                                    ;cmp   [si+0Bh],cx    ; BPB.fatsecs16
35106 000029BF 394C0B                    cmp    [si+A_BPB.SECTORSPERFAT],cx ; 0
35107 000029C2 7514                      jnz    short set_dpb  ; FAT DPB (33 bytes)   -jnz-
35108                                                          ; FAT32 DPB (61 bytes) -jz-
35109                                    ;mov   [es:bp+39h],cx
35110 000029C4 26894E39                  mov    [es:bp+DPB.FAT32_NXTFREE],cx ; 0
35111                                    ;mov   [es:bp+3Bh],cx
35112 000029C8 26894E3B                  mov    [es:bp+DPB.FAT32_NXTFREE+2],cx ; 0
35113 000029CC 49                        dec    cx     ; 0FFFFh ; -1
35114                                    ;mov   [es:bp+1Fh],cx
35115 000029CD 26894E1F                  mov    [es:bp+DPB.FREE_CNT],cx ; -1 = unknown
35116                                    ;mov   [es:bp+21h],cx
35117 000029D1 26894E21                  mov    [es:bp+DPB.FREE_CNT+2],cx ; -1 = unknown
35118 000029D5 B95845                    mov    cx,4558h       ; 'EX' signature for FAT32 extended DPB
35119                                  set_dpb:
35120                                    ;;;
35121
35122 000029D8 B453                      mov    ah,SETDPB ; 53h              ;hidden system call
35123 000029DA CD21                      int    21h
35124                                           ; DOS - 2+ internal - TRANSLATE BIOS PARAMETER BLOCK
35125                                           ; DS:SI -> BPB (BIOS Parameter Block)
35126                                           ; ES:BP -> buffer for DOS Drive Parameter Block
35127                                    ; 13/04/2024
35128                                    ;;;
35129 000029DC 59                        pop    cx
35130 000029DD 5A                        pop    dx
35131                                    ;;;
35132
35133                                    ;mov   ax,[es:bp+2]
35134 000029DE 268B4602                  mov    ax,[es:bp+DPB.SECTOR_SIZE]
35135 000029E2 06                        push   es
35136 000029E3 2EC43E[6D02]              les    di,[cs:DOSINFO]       ;es:di point to dos info
35137                                    ;cmp   ax,[es:di+10h]
35138 000029E8 263B4510                  cmp    ax,[es:di+SYSI_MAXSEC]
35139 000029EC 07                        pop    es
35140                                    ; 13/04/2024
35141                                    ;jna   short iblk_1
35142                                    ;jmp   bad_bpb_size_sector
35143                                    ; 29/10/2022
35144 000029ED 777F                      ja     short bad_bpb_size_sector
35145                                  iblk_1:
35146 000029EF 1E                        push   ds
35147 000029F0 52                        push   dx
35148
35149 000029F1 2EC516[3924]              lds    dx,[cs:DevEntry]
35150                                    ;mov   [es:bp+13h],dx
35151 000029F6 26895613                  mov    [es:bp+DPB.DRIVER_ADDR],dx
35152                                    ;mov   [es:bp+15h],ds
35153 000029FA 268C5E15                  mov    [es:bp+DPB.DRIVER_ADDR+2],ds
35154
35155 000029FE 5A                        pop    dx
35156 000029FF 1F                        pop    ds
35157
35158 00002A00 42                        inc    dx
35159 00002A01 FEC6                      inc    dh
35160                                    ;loop  perunit
35161                                    ; 13/04/2024
35162                                    ;;;
35163 00002A03 49                        dec    cx             ; cx = cx - 1
35164                                                          ; cx = remain count from [cs:unitcount]
35165 00002A04 7403                      jz     short iblk_2   ; cx = 0 -> done
35166 00002A06 E964FF                    jmp    perunit        ; loop until cx is 0
35167                                  iblk_2:
35168                                    ;;;
35169
35170 00002A09 0E                        push   cs
35171 00002A0A 1F                        pop    ds
35172
35173 00002A0B E895E3                    call   TempCDS               ; set cds for new drives
35174                                    ; 31/12/2022
35175                                    ; ds <> cs
35176                                  linkit:
35177 00002A0E 2EC43E[6D02]              les    di,[cs:DOSINFO]       ;es:di = dos table
35178 00002A13 268B4D22                  mov    cx,[es:di+SYSI_DEV]   ;dx:cx = head of list
35179 00002A17 268B5524                  mov    dx,[es:di+SYSI_DEV+2]
35180
```

```
35181 00002A1B 2EC536[3924]               lds     si,[cs:DevEntry]      ;ds:si = device location
35182 00002A20 26897522                   mov     [es:di+SYSI_DEV],si   ;set head of list in dos
35183 00002A24 268C5D24                   mov     [es:di+SYSI_DEV+2],ds
35184 00002A28 8B04                       mov     ax,[si]               ;get pointer to next device
35185 00002A2A 2EA3[3924]                 mov     [cs:DevEntry],ax      ;and save it
35186
35187 00002A2E 890C                       mov     [si],cx               ;link in the driver
35188 00002A30 895402                     mov     [si+2],dx
35189                                   enddev:
35190 00002A33 5E                         pop     si
35191 00002A34 07                         pop     es
35192 00002A35 40                         inc     ax                    ;ax = ffff (no more devs if yes)?
35193 00002A36 740B                       jz      short coffj3
35194
35195 00002A38 2EFE06[6619]               inc     byte [cs:multdeviceflag] ; possibly multiple device driver.
35196 00002A3D E8E80C                     call    DevBreak              ; M009
35197                                     ; 11/12/2022
35198                                     ; ds = cs (DevBreak)
35199
35200                                     ; 03/04/2019 - Retro DOS v4.0
35201                                     ; MSDOS 6.21 IO.SYS - SYSINIT:290Dh
35202 00002A40 E9EDFD                     jmp     goodld                ; otherwise pretend we loaded it in
35203                                   coffj3:
35204                                     ; 18/12/2022
35205                                     ; ax = 0
35206 00002A43 2EA2[6619]                 mov     [cs:multdeviceflag],al ; 0
35207                                     ;mov    byte [cs:multdeviceflag],0 ; reset the flag
35208 00002A47 E8DE0C                     call    DevBreak
35209                                     ; 11/12/2022
35210                                     ; ds = cs (DevBreak)
35211
35212                                     ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
35213                                     ; (SYSINIT:2919h)
35214                                     ; 11/07/2023
35215 00002A4A E80204                     call    CheckProtmanArena
35216
35217                                     ; 02/11/2022 (MSDOS 5.0 IO.SYS compatibility)
35218                                     ;;call  CheckProtmanArena     ; adjust alloclim if Protman$ just
35219                                   ;                               ;  created a bogus arena to try
35220                                   ;                               ;  to protect some of its resident-
35221                                   ;                               ;  init code.
35222                                     ; 13/04/2024 - Retro DOS v5.0
35223                                     ; PCDOS 7.1 IBMBIO.COM
35224                                     ;;call  CheckDoubleSpace
35225                                     ;jmp   coff
35226
35227                                   ;--------------------------------------------------------------------------
35228
35229                                     ; 13/04/2024 - Retro DOS v5.0
35230                                     ; PCDOS 7.1 IBMBIO.COM
35231                                     ;;;
35232
35233                                   CheckDoubleSpace:
35234
35235                                   ;;  ifdef  dblspace_hooks
35236
35237                                   ;   Now check for two special MagicDrv cases:
35238                                   ;
35239                                   ;       a) the last driver load was MagicDrv final placement:
35240                                   ;      -> add number of MagicDrv reserved drives to drivenumber
35241                                   ;
35242                                   ;       b) MagicDrv is currently in temporary home:
35243                                   ;          -> call it to give it a chance to mount and shuffle drives
35244
35245                                     ;cmp   byte [cs:MagicHomeFlag],0 ; already home?
35246                                     ; 13/04/2024 (PCDOS 7.1 IBMBIO.COM)
35247 00002A4D 2EF606[6724]01             test   byte [cs:MagicHomeFlag],1 ; already home?
35248 00002A53 7545                       jnz    short no_more_magic_calls ;  nothing more to do if so
35249
35250                                   ;   Now inquire of driver whether it is present, and final located
35251
35252                                     ;mov   ax,multMagicdrv ; 4A11h
35253                                     ;mov   bx,MD_VERSION ; 0
35254                                     ;int   2fh                    ; ch = number of MagicDrv drive letters
35255                                     ;or    ax,ax                  ; is it there?
35256                                     ; 13/04/2024 (PCDOS 7.1 IBMBIO.COM)
35257                                     ;;;
35258 00002A55 E8FEEE                     call   get_dblspace_version ; is it there?
35259                                     ;jnz   short no_more_magic_calls ; done if not
35260 00002A58 750B                       jnz    short set_magichomeflag
35261                                     ;;;
35262
35263 00002A5A F7C20080                   test   dx,8000h               ; is it final placed?
35264 00002A5E 751C                       jnz    short magic_not_yet_home ;  skip if not
35265
35266                                   ;   Okay, now the driver is final placed!  Set the flag so we
35267                                   ;   don't keep checking it, and add its number of drive letters
35268                                   ;   to drivenumber.
35269
35270                                     ; 13/04/2024 (PCDOS 7.1 IBMBIO.COM)
35271                                     ;mov   byte [cs:MagicHomeFlag],0ffh ; set the flag!
35272 00002A60 2E002E[8503]                add    [cs:drivenumber],ch    ; add number of MagicDrv volumes to
35273                                                                     ;  the drive number we'll pass to the
35274                                                                     ;  next loadable block device.
35275                                     ;jmp   short no_more_magic_calls ; and finished.
35276
35277                                     ;;;
35278                                   set_magichomeflag:
35279 00002A65 2EC606[6724]01             mov    byte [cs:MagicHomeFlag],1 ; set the flag!
35280 00002A6B E918FB                     jmp    coff
35281                                     ;;;
35282
35283                                   ; 03/04/2019 - Retro DOS v4.0
35284
35285                                   bad_bpb_size_sector:
35286 00002A6E 5E                         pop    si
35287 00002A6F 07                         pop    es
35288 00002A70 BA[9250]                   mov    dx,badsiz_pre
35289 00002A73 BB[7050]                   mov    bx,crlfm
35290 00002A76 E8B91F                     call   prnerr
35291
35292 00002A79 E90AFB                     jmp    coff
35293
35294                                   magic_not_yet_home:
35295 00002A7C 06                         push   es
35296 00002A7D 56                         push   si
35297
35298 00002A7E 2E8B0E[6403]               mov    cx,[cs:memhi]          ; pass it a work buffer
35299 00002A83 2E8B16[A502]               mov    dx,[cs:ALLOCLIM]       ;   address in cx (segment)
35300 00002A88 29CA                       sub    dx,cx                  ;   for len dx (paragraphs)
35301
35302 00002A8A BB0200                     mov    bx,2
35303 00002A8D 2EA0[6A19]                 mov    al,[cs:driver_units]   ; shuffle magicdrives and new drives
35304                                                                   ;   by this many units
```

```
                                ;BUGBUG 29-Oct-1992 bens Take this 55h out after Beta 4
00002A91 B455                       mov     ah,55h                  ; backdoor won't shuffle unless it
                                                                    ;  sees this, to prevent bad things
                                                                    ;  from happening if people run the
                                                                    ;  new driver with an old BIOS
00002A93 2EFF1E[9003]               call    far [cs:MagicBackdoor]

00002A98 5E                         pop     si
00002A99 07                         pop     es

                                ;no_more_magic_calls:
                                ;
                                ;;  endif
                                ;   retn

                                    ; 13/04/2024
                                    ;;;
                                no_more_magic_calls:
00002A9A E9E9FA                     jmp     coff
                                    ;;;

                                ;--------------------------------------------------------------------------------
                                ; country command
                                ;      the syntax is:
                                ;   country=country id {,codepage {,path}}
                                ;   country=country id {,,path}   :default codepage id in dos
                                ;--------------------------------------------------------------------------------

                                    ; 30/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:2663h)
                                tryq:
00002A9D 80FC51                         cmp     ah,CONFIG_COUNTRY ; 'Q'
00002AA0 7403                       je      short tryq_cont
                                skip_it3:
00002AA2 E90D01                     jmp     tryf
                                tryq_cont:

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:297Eh)
                                ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                ;ifdef      MULTI_CONFIG
00002AA5 E8741B                     call    query_user              ; query the user if config_cmd
00002AA8 72F8                       jc      short skip_it3          ; has the CONFIG_OPTION_QUERY bit set
                                ;endif
                                ;%endif ; 02/11/2022

                                    ; 31/12/2022
                                    ;xor    bx,bx
00002AAA 31C9                       xor     cx,cx
                                    ; 14/12/2022
                                    ; ds = cs
                                    ; bx = 0
                                    ;mov    byte [cs:cntry_drv],0 ; reset the drive,path to default value.
                                    ;mov    word [cs:p_code_page],0
                                    ; 31/12/2022
                                    ; cx = 0
                                    ;mov    [cntry_drv],bl ; 0
                                    ;mov    [p_code_page],bx ; 0
00002AAC 880E[DD4A]                 mov     [cntry_drv],cl ; 0
00002AB0 890E[7C22]                 mov     [p_code_page],cx ; 0

00002AB4 BF[4522]                   mov     di,cntry_parms
                                    ;xor    cx,cx   ; 31/12/2022
                                    ; 03/01/2023
                                    ;mov    dx,cx
                                do52:
00002AB7 E8AD03                     call    sysinit_parse
00002ABA 730B                       jnc     short if52              ; parse error,check error code and

00002ABC E8E000                     call    cntry_error             ; show message and end the search loop.
                                    ; 14/12/2022
                                    ; ds = cs
00002ABF C706[7A22]FFFF             mov     word [p_cntry_code],-1
                                    ;mov    word [cs:p_cntry_code],-1 ; signals that parse error.
00002AC5 EB34                       jmp     short sr52
                                if52:
00002AC7 83F8FF                     cmp     ax,_$P_RC_EOL ; 0FFFFh; end of line?
00002ACA 742F                       jz      short sr52              ; then end the search loop

                                    ;cmp    byte [cs:result_val+_$P_Result_Blk.Type],_$P_number ; numeric?
                                    ; 14/12/2022
                                    ; ds = cs
00002ACC 803E[1722]01               cmp     byte [result_val],_$P_Number
                                    ;cmp    byte [cs:result_val],_$P_Number
00002AD1 7512                       jnz     short if56

                                    ;;mov   ax,[cs:rw_dword]
                                    ;mov    ax,[cs:result_val+_$P_Result_Blk.Picked_Val]
                                    ; 14/12/2022
00002AD3 A1[1B22]                   mov     ax,[result_val+_$P_Result_Blk.Picked_Val]
00002AD6 83F901                     cmp     cx,1
00002AD9 7505                       jne     short if57

                                    ;mov    [cs:p_cntry_code],ax
                                    ; 14/12/2022
00002ADB A3[7A22]                   mov     [p_cntry_code],ax

                                    ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;jmp    short en57
                                    ; 12/12/2022
                                    ;jmp    short en56
00002ADE EBD7                       jmp     short do52
                                if57:
                                    ;mov    [cs:p_code_page],ax
                                    ; 14/12/2022
                                    ; ds = cs
00002AE0 A3[7C22]                   mov     [p_code_page],ax
                                en57:
                                    ;jmp    short en56              ; path entered
                                    ; 12/12/2022
00002AE3 EBD2                       jmp     short do52
                                if56:
00002AE5 1E                         push    ds
00002AE6 06                         push    es
00002AE7 56                         push    si
00002AE8 57                         push    di

00002AE9 0E                         push    cs
00002AEA 07                         pop     es

                                    ;lds    si,[cs:rv_dword]        ; move the path to known place.
                                    ; 14/12/2022
```

```
35429 00002AEB C536[1B22]               lds     si,[rv_dword]
35430 00002AEF BF[DD4A]                 mov     di,cntry_drv
35431 00002AF2 E82C1F                   call    move_asciiz
35432
35433 00002AF5 5F                       pop     di
35434 00002AF6 5E                       pop     si
35435 00002AF7 07                       pop     es
35436 00002AF8 1F                       pop     ds
35437                                 en56:
35438 00002AF9 EBBC                     jmp     short do52
35439                                 sr52:
35440                                   ; 14/12/2022
35441                                   ; ds = cs
35442 00002AFB 833E[7A22]FF             cmp     word [p_cntry_code],-1
35443                                   ;cmp    word [cs:p_cntry_code],-1     ; had a parse error?
35444 00002B00 7509                     jne     short tryq_open
35445 00002B02 E981FA                   jmp     coff
35446
35447                                 tryqbad:                        ;"invalid country code or code page"
35448 00002B05 F9                       stc
35449 00002B06 BA[D950]                 mov      dx,badcountry
35450 00002B09 EB79                     jmp      tryqchkerr
35451
35452                                 tryq_open:
35453                                   ; 14/12/2022
35454                                   ; ds = cs
35455 00002B0B 803E[DD4A]00             cmp     byte [cntry_drv],0
35456                                   ;cmp    byte [cs:cntry_drv],0
35457 00002B10 7405                     je      short tryq_def
35458 00002B12 BA[DD4A]                 mov     dx,cntry_drv
35459 00002B15 EB03                     jmp     short tryq_openit
35460
35461                                 tryq_def:
35462 00002B17 BA[DF4A]                 mov     dx,cntry_root
35463                                 tryq_openit:
35464 00002B1A B8003D                   mov     ax,3D00h                ;open a file
35465 00002B1D F9                       stc
35466 00002B1E CD21                     int     21h
35467 00002B20 7242                     jc      short tryqfilebad       ;open failure
35468
35469                                   ; 14/12/2022
35470                                   ; ds = cs
35471 00002B22 A3[5C03]                 mov     [cntryfilehandle],ax
35472                                   ;mov    [cs:cntryfilehandle],ax       ;save file handle
35473 00002B25 89C3                     mov     bx,ax
35474 00002B27 A1[7A22]                 mov     ax,[p_cntry_code]
35475 00002B2A 8B16[7C22]               mov     dx,[p_code_page]
35476                                   ;mov    ax,[cs:p_cntry_code]
35477                                   ;mov    dx,[cs:p_code_page]   ;now,ax=country id,bx=filehandle
35478                                   ;mov    cx,[cs:memhi]
35479 00002B2E 8B0E[6403]               mov     cx,[memhi]
35480 00002B32 81C18001                 add     cx,384                  ;need 6k buffer to handle country.sys
35481                                                                   ;MO23
35482                                   ; 14/12/2022
35483                                   ; ds = cs
35484 00002B36 3B0E[A502]               cmp     cx,[ALLOCLIM]
35485                                   ;cmp    cx,[cs:ALLOCLIM]
35486 00002B3A 7745                     ja      short tryqmemory        ;cannot allocate the buffer for country.sys
35487
35488 00002B3C BE[DD4A]                 mov     si,cntry_drv            ;ds:si -> cntry_drv
35489 00002B3F 803C00                   cmp     byte [si],0             ;default path?
35490 00002B42 7502                     jne     short tryq_set_for_dos
35491
35492 00002B44 46                       inc     si
35493 00002B45 46                       inc     si                      ;ds:si -> cntry_root
35494
35495                                 tryq_set_for_dos:
35496                                   ; 14/12/2022
35497                                   ; ds = cs
35498 00002B46 C43E[7902]               les     di,[sysi_country]
35499                                   ;les    di,[cs:sysi_country]  ;es:di -> country info tab in dos
35500 00002B4A 57                       push    di                      ;save di
35501                                   ;add    di,8
35502 00002B4B 83C708                   add     di,country_cdpg_info.ccPath_CountrySys ; 8
35503 00002B4E E8D01E                   call    move_asciiz             ;set the path to country.sys in dos.
35504 00002B51 5F                       pop     di                      ;es:di -> country info tab again.
35505
35506                                   ; 14/12/2022
35507 00002B52 8B0E[6403]               mov     cx,[memhi]
35508                                   ;mov    cx,[cs:memhi]
35509 00002B56 8ED9                     mov     ds,cx
35510 00002B58 31F6                     xor     si,si                   ;ds:si -> 2k buffer to be used.
35511 00002B5A E8601D                   call    setdoscountryinfo       ;now do the job!!!
35512                                   ; ds <> cs ; 14/12/2022
35513 00002B5D 7325                     jnc     short tryqchkerr        ;read error or could not find country,code page combination
35514
35515 00002B5F 83F9FF                   cmp     cx,-1                   ;could not find matching country_id,code page?
35516 00002B62 74A1                     je      short tryqbad           ;then "invalid country code or code page"
35517
35518                                 tryqfilebad:
35519 00002B64 0E                       push    cs
35520 00002B65 07                       pop     es
35521 00002B66 2E803E[DD4A]00           cmp     byte [cs:cntry_drv],0 ;is the default file used?
35522 00002B6C 7405                     je      short tryqdefbad
35523
35524 00002B6E BE[DD4A]                 mov     si,cntry_drv
35525 00002B71 EB03                     jmp     short tryqbadload
35526
35527                                 tryqdefbad:                     ;default file has been used.
35528 00002B73 BE[DF4A]                 mov     si,cntry_root           ;es:si -> \country.sys in sysinit_seg
35529                                 tryqbadload:
35530 00002B76 E8B31E                   call    badload                 ;ds will be restored to sysinit_seg
35531                                   ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
35532                                   ; (SYSINIT:2A69h)
35533 00002B79 8B0E[A302]               mov     cx,[CONFBOT] ; ds = cs (from badload)
35534                                   ;mov    cx,[cs:CONFBOT]
35535                                   ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
35536                                   ;mov    cx,[cs:top_of_cdss]
35537                                   ; 11/12/2022
35538                                   ; ds = cs
35539                                   ;mov    cx,[top_of_cdss]  ; mov cx,[CONFBOT]
35540 00002B7D 8EC1                     mov     es,cx                   ;restore es -> confbot.
35541 00002B7F EB13                     jmp     short coffj4
35542
35543                                 tryqmemory:
35544 00002B81 BA[1C51]                 mov     dx,insufmemory
35545                                 tryqchkerr:
35546                                   ;mov    cx,[cs:CONFBOT]
35547                                   ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
35548                                   ;mov    cx,[cs:top_of_cdss]
35549                                   ; 12/12/2022
35550 00002B84 0E                       push    cs
35551 00002B85 1F                       pop     ds
35552                                   ; 31/12/2022 - Retro DOS v4.2
```

```
00002B86 8B0E[A302]                 mov     cx,[CONFBOT] ; (MSDOS 6.21 IO.SYS, SYSINIT)
                                    ;mov    cx,[top_of_cdss]  ; mov cx,[CONFBOT]
00002B8A 8EC1                       mov     es,cx                   ;restore es -> confbot seg
                                    ;push   cs
                                    ;pop    ds                      ;restore ds to sysinit_seg
00002B8C 7306                       jnc     short coffj4            ;if no error,then exit

00002B8E E8C31E                     call    print                   ;else show error message
00002B91 E82603                     call    error_line
                                coffj4:
                                    ;mov    bx,[cs:cntryfilehandle]
                                    ; 11/12/2022
                                    ; ds = cs
00002B94 8B1E[5C03]                 mov     bx,[cntryfilehandle]
00002B98 B43E                       mov     ah,3Eh
00002B9A CD21                       int     21h                     ;close a file. don't care even if it fails.
00002B9C E9E7F9                     jmp     coff

                                ;-----------------------------------------

                                cntry_error:

                                ;function: show "invalid country code or code page" messages,or
                                ;          "error in country command" depending on the error code
                                ;          in ax returned by sysparse;
                                ;in:ax - error code
                                ;   ds - sysinitseg
                                ;   es - confbot
                                ;out:      show message.  dx destroyed.

00002B9F 83F806                     cmp     ax,_$P_Out_Of_Range ; 6
00002BA2 7505                       jne     short if64
00002BA4 BA[D950]                   mov     dx,badcountry           ;"invalid country code or code page"
00002BA7 EB03                       jmp     short en64
                                if64:
00002BA9 BA[FF50]                   mov     dx,badcountrycom        ;"error in contry command"
                                en64:
00002BAC E8A51E                     call    print
                                    ;call   error_line
                                    ;retn
                                    ; 11/12/2022
00002BAF E90803                     jmp     error_line

                                ;---------------------------------------------------------------------------
                                ; files command
                                ;---------------------------------------------------------------------------

                                ;****************************************************************************
                                ; function: parse the parameters of files= command.                         *
                                ;                                                                          *
                                ; input :                                                                  *
                                ;   es:si -> parameters in command line.                                   *
                                ; output:                                                                 *
                                ;   variable files set.                                                    *
                                ;                                                                          *
                                ; subroutines to be called:                                               *
                                ;   sysinit_parse                                                          *
                                ; logic:                                                                  *
                                ; {                                                                        *
                                ;   set di points to files_parms;                                          *
                                ;   set dx,cx to 0;                                                        *
                                ;   while (end of command line)                                            *
                                ;   { sysinit_parse;                                                       *
                                ;     if (no error) then                                                   *
                                ;        files = result_val._$P_picked_val                                 *
                                ;     else                                                                 *
                                ;        error exit;                                                       *
                                ;   };                                                                     *
                                ; };                                                                      *
                                ;                                                                          *
                                ;****************************************************************************

                                tryf:
00002BB2 80FC46                         cmp     ah,CONFIG_FILES ;  'F'
00002BB5 7528                       jne     short tryl

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:2AABh)
                                ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                ;ifdef     MULTI_CONFIG
00002BB7 E8621A                     call    query_user              ; query the user if config_cmd
00002BBA 7223                       jc      short tryl              ; has the CONFIG_OPTION_QUERY bit set
                                ;endif
                                ;%endif ; 30/10/2022

                                    ; 14/12/2022
                                    ; ds = cs

00002BBC BF[7E22]                   mov     di,files_parms
00002BBF 31C9                       xor     cx,cx
                                    ; 03/01/2023
                                    ;mov    dx,cx
                                do67:
00002BC1 E8A302                     call    sysinit_parse
00002BC4 7303                       jnc     short if67              ; parse error
                                    ;call   badparm_p               ;  and show messages and end the search loop.
                                    ;jmp    short sr67
                                    ; 03/01/2023
00002BC6 E98D01                     jmp     badparm_p_coff
                                if67:
00002BC9 83F8FF                     cmp     ax,_$P_RC_EOL           ; end of line?
00002BCC 7408                       je      short en67              ; then end the $endloop

                                    ; 14/12/2022
                                    ; ds = cs
                                    ;;mov   al,[cs:rv_dword]
                                    ;mov    al,[cs:result_val+_$P_Result_Blk.Picked_Val]
                                    ;mov    [cs:p_files],al                 ; save it temporarily
                                    ;mov    al,[rv_dword]
00002BCE A0[1B22]                   mov     al,[result_val+_$P_Result_Blk.Picked_Val]
00002BD1 A2[9D22]                   mov     [p_files],al

00002BD4 EBEB                       jmp     short do67
                                en67:
                                    ; 14/12/2022
                                    ; ds = cs
00002BD6 A0[9D22]                   mov     al,[p_files]
00002BD9 A2[9F02]                   mov     [FILES],al
                                    ;mov    al,[cs:p_files]
                                    ;mov    [cs:FILES],al           ; no error. really set the value now.
                                sr67:
00002BDC E9A7F9                     jmp     coff
```

```
                                ; 04/04/2019 - Retro DOS v4.0

                                ;--------------------------------------------------------------------------
                                ; lastdrive command
                                ;--------------------------------------------------------------------------

                                ;**************************************************************************
                                ; function: parse the parameters of lastdrive= command.                    *
                                ;                                                                         *
                                ; input :                                                                 *
                                ;   es:si -> parameters in command line.                                  *
                                ; output:                                                                *
                                ;   set the variable num_cds.                                           *
                                ;                                                                         *
                                ; subroutines to be called:                                              *
                                ;   sysinit_parse                                                       *
                                ; logic:                                                                 *
                                ; {                                                                       *
                                ;   set di points to ldrv_parms;                                        *
                                ;   set dx,cx to 0;                                                     *
                                ;   while (end of command line)                                         *
                                ;   { sysinit_parse;                                                    *
                                ;     if (no error) then                                                *
                                ;        set num_cds to the returned value;                             *
                                ;     else /*error exit*/                                               *
                                ;        error exit;                                                    *
                                ;   };                                                                  *
                                ; };                                                                     *
                                ;                                                                       *
                                ;**************************************************************************

                                tryl:
00002BDF 80FC4C                         cmp     ah,CONFIG_LASTDRIVE ; 'L'
00002BE2 7528                       jne     short tryp

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:2AE0h)
                                ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
00002BE4 E8351A                     call    query_user      ; query the user if config_cmd
00002BE7 7223                       jc      short tryp      ; has the CONFIG_OPTION_QUERY bit set
                                ;endif
                                ;%endif ; 30/10/2022

                                    ; 14/12/2022
                                    ; ds = cs

00002BE9 BF[D522]                   mov     di,ldrv_parms
00002BEC 31C9                       xor     cx,cx
                                    ; 03/01/2023
                                    ;mov    dx,cx
                                do73:
00002BEE E87602                     call    sysinit_parse
00002BF1 7303                       jnc     short if73      ; parse error
                                    ;call   badparm_p       ;  and show messages and end the search loop.
                                    ;jmp    short sr73
                                    ; 03/01/2023
00002BF3 E96001                     jmp     badparm_p_coff
                                if73:
00002BF6 83F8FF                     cmp     ax,_$P_RC_EOL  ; end of line?
00002BF9 7408                       je      short en73     ; then end the $endloop

                                    ; 14/12/2022
                                    ; ds = cs
                                    ;;mov   al,[cs:rv_dword]
                                    ;mov    al,[cs:rv_byte]       ; pick up the drive number
                                    ;mov    [cs:p_ldrv],al ; save it temporarily

                                    ;mov    al,[rv_dword]
00002BFB A0[1B22]                   mov     al,[rv_byte]
00002BFE A2[E922]                   mov     [p_ldrv],al

00002C01 EBEB                       jmp     short do73
                                en73:
                                    ; 14/12/2022
                                    ; ds = cs
00002C03 A0[E922]                   mov     al,[p_ldrv]
00002C06 A2[A202]                   mov     [NUM_CDS],al
                                    ;mov    al,[cs:p_ldrv]
                                    ;mov    [cs:NUM_CDS],al       ; no error. really set the value now.
                                sr73:
00002C09 E97AF9                     jmp     coff

                                ;--------------------------------------------------------------------------
                                ; setting drive parameters
                                ;--------------------------------------------------------------------------

                                tryp:
00002C0C 80FC50                         cmp     ah,CONFIG_DRIVPARM ; 'P'
00002C0F 7516                       jne     short tryk

                                ; 31/12/2022 - Retro DOS v4.2
                                ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                ;ifdef      MULTI_CONFIG
00002C11 E8081A                     call    query_user      ; query the user if config_cmd
00002C14 7211                       jc      short tryk      ; has the CONFIG_OPTION_QUERY bit set
                                ;endif
                                ;%endif ; 30/10/2022

00002C16 E8EC0E                     call    parseline
00002C19 7209                       jc      short trypbad
00002C1B E8050E                     call    setparms
00002C1E E8470E                     call    diddleback

                                ; No error check here, because setparms and diddleback have no error
                                ; returns, and setparms as coded now can return with carry set.
                                ;       jc  short trypbad

                                    ; 12/12/2022
                                    ; cf = 0
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;jc     short trypbad

00002C21 E962F9                     jmp     coff
                                trypbad:
00002C24 E95F02                     jmp     badop

                                ;--------------------------------------------------------------------------
                                ; setting internal stack parameters
                                ; stacks=m,n where
                                ;   m is the number of stacks (range 8 to 64,default 9)
                                ;   n is the stack size (range 32 to 512 bytes,default 128)
                                ; j.k. 5/5/86: stacks=0,0 implies no stack installation.
```

```
35801                                   ;   any combinations that are not within the specified limits will
35802                                   ;   result in "unrecognized command" error.
35803                                   ;-----------------------------------------------------------------------
35804
35805                                   ;***********************************************************************
35806                                   ;                                                                      *
35807                                   ; function: parse the parameters of stacks= command.                   *
35808                                   ;       the minimum value for "number of stacks" and "stack size" is  *
35809                                   ;       8 and 32 each.  in the definition of sysparse value list,they *
35810                                   ;       are set to 0.  this is for accepting the exceptional case of  *
35811                                   ;       stacks=0,0 case (,which means do not install the stack.)       *
35812                                   ;       so,after sysparse is done,we have to check if the entered      *
35813                                   ;       values (stack_count,stack_size) are within the actual range,   *
35814                                   ;       (or if "0,0" pair has been entered.)                           *
35815                                   ; input :                                                              *
35816                                   ;   es:si -> parameters in command line.                              *
35817                                   ; output:                                                              *
35818                                   ;   set the variables stack_count,stack_size.                        *
35819                                   ;                                                                      *
35820                                   ; subroutines to be called:                                            *
35821                                   ;   sysinit_parse                                                     *
35822                                   ; logic:                                                               *
35823                                   ; {                                                                    *
35824                                   ;   set di points to stks_parms;                                      *
35825                                   ;   set dx,cx to 0;                                                   *
35826                                   ;   while (end of command line)                                       *
35827                                   ;   { sysinit_parse;                                                  *
35828                                   ;     if (no error) then                                               *
35829                                   ;        { if (cx == 1) then /* first positional = stack count */      *
35830                                   ;             p_stack_count = result_val._$P_picked_val;            *
35831                                   ;          if (cx == 2) then /* second positional = stack size */     *
35832                                   ;             p_stack_size = result_val._$P_picked_val;             *
35833                                   ;        }                                                             *
35834                                   ;     else /*error exit*/                                              *
35835                                   ;        error exit;                                                   *
35836                                   ;   };                                                                 *
35837                                   ;   here check p_stack_count,p_stack_size if it meets the condition;  *
35838                                   ;   if o.k.,then set stack_count,stack_size;                          *
35839                                   ;    else error_exit;                                                  *
35840                                   ; };                                                                   *
35841                                   ;***********************************************************************
35842
35843                                   tryk:
35844                                           ;if      stacksw
35845
35846 00002C27 80FC4B                         cmp     ah,CONFIG_STACKS ; 'K'
35847 00002C2A 7402                      je      short do_tryk
35848                                   skip_it4:
35849 00002C2C EB79                      jmp     short trys     ; 15/12/2022
35850                                   do_tryk:
35851
35852                                   ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
35853                                   ; (SYSINIT:2B33h)
35854                                   ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
35855                                   ;%if 0
35856                                   ;ifdef      MULTI_CONFIG
35857 00002C2E E8EB19                         call query_user              ; query the user if config_cmd
35858 00002C31 72F9                           jc   short skip_it4        ; has the CONFIG_OPTION_QUERY bit set
35859                                   ;endif
35860                                   ;%endif     ; 30/10/2022
35861
35862                                       ; 14/12/2022
35863                                       ; ds = cs
35864
35865 00002C33 BF[EA22]                     mov     di,stks_parms
35866 00002C36 31C9                         xor     cx,cx
35867                                       ; 03/01/2023
35868                                       ;mov    dx,cx
35869                                   do79:
35870 00002C38 E82C02                       call    sysinit_parse
35871 00002C3B 730B                         jnc     short if79             ; parse error
35872
35873 00002C3D BA[8B51]                     mov     dx,badstack            ; "invalid stack parameter"
35874 00002C40 E8111E                       call    print                  ;  and show messages and end the search loop.
35875 00002C43 E87402                       call    error_line
35876                                       ;jmp    sr79
35877                                       ; 11/12/2022
35878 00002C46 EB39                         jmp     short sr79
35879                                   if79:
35880 00002C48 83F8FF                       cmp     ax,_$P_RC_EOL          ; end of line?
35881 00002C4B 7412                         je      short en79             ; then end the $endloop
35882
35883                                       ; 14/12/2022
35884                                       ; ds = cs
35885
35886                                       ;;mov   ax,[cs:rv_dword]
35887                                       ;mov    ax,[cs:result_val+_$P_Result_Blk.Picked_Val]
35888                                       ;mov    ax,[rv_dword]
35889 00002C4D A1[1B22]                     mov     ax,[result_val+_$P_Result_Blk.Picked_Val]
35890
35891 00002C50 83F901                       cmp     cx,1
35892 00002C53 7505                         jne     short if83
35893
35894                                       ; 14/12/2022
35895                                       ;mov    [cs:p_stack_count],ax
35896                                       ;jmp    short en83
35897 00002C55 A3[1F23]                     mov     [p_stack_count],ax
35898 00002C58 EBDE                         jmp     short do79
35899                                   if83:
35900                                       ; 14/12/2022
35901                                       ;mov    [cs:p_stack_size],ax
35902 00002C5A A3[2123]                     mov     [p_stack_size],ax
35903                                   en83:
35904 00002C5D EBD9                         jmp     short do79
35905                                   en79:
35906                                       ; 14/12/2022
35907                                       ; ds = cs
35908 00002C5F A1[1F23]                     mov     ax,[p_stack_count]
35909 00002C62 09C0                         or      ax,ax
35910 00002C64 741E                         jz      short if87
35911
35912                                       ; 14/12/2022
35913                                       ;cmp    word [p_stack_count],0
35914                                       ;;cmp   word [cs:p_stack_count],0
35915                                       ;je     short if87
35916
35917                                       ; 14/12/2022
35918 00002C66 83F808                       cmp     ax, mincount ; 8
35919                                       ;cmp    word [cs:p_stack_count],mincount ; 8
35920                                       ; 15/12/2022
35921 00002C69 721F                         jb      short en87
35922 00002C6B 833E[2123]20                 cmp     word [p_stack_size],minsize ; 32
35923                                       ;cmp    word [cs:p_stack_size],minsize ; 32
35924                                       ; 15/12/2022
```

```
35925 00002C70 7218                    jb      short en87
35926                                  if94:
35927                                      ; 14/12/2022
35928                                      ; ds = cs
35929                                      ; ax = [p_stack_count]
35930                                      ;mov    ax,[p_stack_count]
35931                                      ;;mov   ax,[cs:p_stack_count]
35932 00002C72 A3[8C02]                   mov     [stack_count],ax
35933                                      ;mov    [cs:stack_count],ax
35934                                      ;mov    ax,[cs:p_stack_size]
35935 00002C75 A1[2123]                   mov     ax,[p_stack_size]
35936                                      ;mov    [cs:stack_size],ax
35937 00002C78 A3[8E02]                   mov     [stack_size],ax
35938                                      ;mov    word [cs:stack_addr],-1      ; stacks= been accepted.
35939 00002C7B C706[9002]FFFF             mov     word [stack_addr],-1
35940                                  sr79:
35941 00002C81 E902F9                     jmp     coff
35942
35943                                  if87:
35944                                      ; 14/12/2022
35945 00002C84 3906[2123]                 cmp     [p_stack_size],ax ; 0
35946 00002C88 74E8                       je      short if94 ; ax = [p_stack_count] = 0
35947                                      ;cmp    word [cs:p_stack_size],0
35948                                      ;je     short if94
35949                                  en87:
35950                                      ; 15/12/2022
35951                                      ; ([p_stack_count] is invalid, use default values)
35952                                      ; 14/12/2022
35953                                      ; ds = cs
35954 00002C8A C706[8C02]0900             mov     word [stack_count],defaultcount ; 9
35955 00002C90 C706[8E02]8000             mov     word [stack_size],defaultsize ; 128
35956 00002C96 C706[9002]0000             mov     word [stack_addr],0
35957                                      ;mov    word [cs:stack_count],defaultcount ; 9
35958                                      ;                                  ; reset to default value.
35959                                      ;mov    word [cs:stack_size],defaultsize ; 128
35960                                      ;mov    word [cs:stack_addr],0
35961
35962 00002C9C BA[8B51]                    mov     dx,badstack
35963 00002C9F E8B21D                     call    print
35964 00002CA2 E81502                     call    error_line
35965 00002CA5 EBDA                       jmp     short sr79
35966
35967                                  ; 15/12/2022
35968                                  %if 0
35969                                      mov     di,stks_parms
35970                                      xor     cx,cx
35971                                      ; 03/01/2023
35972                                      ;mov    dx,cx
35973                                  do79:
35974                                      call    sysinit_parse
35975                                      jnc     short if79             ; parse error
35976
35977                                      mov     dx,badstack            ; "invalid stack parameter"
35978                                      call    print                  ;  and show messages and end the search loop.
35979                                      call    error_line
35980                                      ;jmp    sr79
35981                                      ; 11/12/2022
35982                                      jmp     short sr79
35983                                  if79:
35984                                      cmp     ax,_$P_RC_EOL          ; end of line?
35985                                      je      short en79             ; then end the $endloop
35986
35987                                      ;mov    ax,[cs:rv_dword]
35988                                      mov     ax,[cs:result_val+_$P_Result_Blk.Picked_Val]
35989                                      cmp     cx,1
35990                                      jne     short if83
35991
35992                                      mov     [cs:p_stack_count],ax
35993                                      jmp     short en83
35994                                  if83:
35995                                      mov     [cs:p_stack_size],ax
35996                                  en83:
35997                                      jmp     short do79
35998                                  en79:
35999                                      cmp     word [cs:p_stack_count],0
36000                                      je      short if87
36001
36002                                      cmp     word [cs:p_stack_count],mincount ; 8
36003                                      jb      short ll88
36004                                      cmp     word [cs:p_stack_size],minsize ; 32
36005                                      jnb     short if88
36006                                  ll88:
36007                                      mov     word [cs:p_stack_count],-1 ; invalid
36008                                  if88:
36009                                      jmp     short en87
36010
36011                                      ; 11/12/2022
36012                                  if94:
36013                                      mov     ax,[cs:p_stack_count]
36014                                      mov     [cs:stack_count],ax
36015                                      mov     ax,[cs:p_stack_size]
36016                                      mov     [cs:stack_size],ax
36017                                      mov     word [cs:stack_addr],-1      ; stacks= been accepted.
36018                                  sr79:
36019                                      jmp     coff
36020
36021                                  if87:
36022                                      cmp     word [cs:p_stack_size],0
36023                                      je      short en87
36024                                      mov     word [cs:p_stack_count],-1 ; invalid
36025                                  en87:
36026                                      cmp     word [cs:p_stack_count],-1 ; invalid?
36027                                      jne     short if94
36028
36029                                      mov     word [cs:stack_count],defaultcount ; 9
36030                                                                     ; reset to default value.
36031                                      mov     word [cs:stack_size],defaultsize ; 128
36032                                      mov     word [cs:stack_addr],0
36033
36034                                      mov     dx,badstack
36035                                      call    print
36036                                      call    error_line
36037                                      jmp     short sr79
36038
36039                                  %endif
36040
36041                                  ; 11/12/2022
36042                                  %if 0
36043                                  if94:
36044                                      mov     ax,[cs:p_stack_count]
36045                                      mov     [cs:stack_count],ax
36046                                      mov     ax,[cs:p_stack_size]
36047                                      mov     [cs:stack_size],ax
36048                                      mov     word [cs:stack_addr],-1      ; stacks= been accepted.
```

```
                                sr79:
                                    jmp     coff
                                %endif
                                    ;endif

                                ;--------------------------------------------------------------------------
                                ; shell command
                                ;--------------------------------------------------------------------------

                                trys:
00002CA7 80FC53                         cmp     ah,CONFIG_SHELL ; 'S'
00002CAA 755A                       jne     short tryx

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:2BE1h)
                                ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                ;ifdef      MULTI_CONFIG
00002CAC E86D19                     call    query_user              ; query the user if config_cmd
00002CAF 7255                       jc      short tryx              ; has the CONFIG_OPTION_QUERY bit set
                                    ; 14/04/2024
                                    ; ds = cs
                                    ;mov    byte [cs:newcmd],1
00002CB1 C606[2A4B]01               mov     byte [newcmd],1
                                ;endif
                                ;%endif ; 30/10/2022

                                    ;;mov   word [cs:command_line],0 ; zap length,first byte of command-line
                                    ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;mov    byte [cs:command_line+1],0
                                    ; 15/12/2022
                                    ; ds = cs
                                    ; 08/09/2023
                                    ;mov    byte [command_line+1],0
00002CB6 C706[BB4B]0000             mov     word [command_line],0 ; zap length,first byte of command-line

00002CBC BF[2E4B]                       mov     di,commnd+1             ; we already have the first char
00002CBF 8845FF                         mov     [di-1],al               ; of the new shell in AL, save it now
                                storeshell:
00002CC2 E8EA1A                     call    getchr
00002CC5 08C0                           or      al,al                   ; this is the normal case: "organize"
00002CC7 741C                           jz  short getshparms        ; put a ZERO right after the filename

00002CC9 3C20                           cmp     al," "                  ; this may happen if there are no args
00002CCB 7209                           jb  short endofshell        ; I suppose...
00002CCD 8805                       mov     [di],al
00002CCF 47                         inc     di
                                        ;cmp    di,commnd+63            ; this makes sure we don't overflow
                                        ;jb short storeshell        ; commnd (the filename)
                                        ;jmp       short endofshell
                                    ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;jmp    short storeshell
                                    ; 03/01/2023
00002CD0 81FF[6C4B]                 cmp     di,commnd+63            ; this makes sure we don't overflow
00002CD4 72EC                           jb  short storeshell        ; commnd (the filename)
                                    ;jmp    short endofshell

                                ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;getshparms:
                                ;   mov     byte [di],0             ; zero-terminate the filename
                                ;   mov     di,command_line+1       ; prepare to process the command-line
                                ;
                                ;parmloop:
                                ;   call    getchr
                                ;   cmp     al," "
                                ;   jb      short endofparms
                                ;   mov     [di],al
                                ;   inc     di
                                ;   cmp     di,command_line+126
                                ;   jb      short parmloop
                                ;endofparms:
                                ;   mov     cx,di
                                ;   sub     cx,command_line+1
                                ;   mov     [cs:command_line],cl
                                ;
                                ;endofshell:
                                ;   mov     byte [di],0             ; zero-terminate the filename (or
                                ;                                   ; the command-line as the case may be)
                                ;skipline:
                                ;       cmp     al,lf       ; 0Ah           ; the safest way to eat the rest of
                                ;       je  short endofline             ; the line: watch for ever-present LF
                                ;call       getchr
                                ;       jnc short skipline          ; keep it up as long as there are chars
                                ;
                                ;endofline:
                                ;       jmp     conflp
                                    ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                endofshell:
00002CD6 C60500                             mov     byte [di],0             ; zero-terminate the filename (or
                                                                        ; the command-line as the case may be)
                                    ; 11/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                    ; MSDOS 6.21 IO.SYS - SYSINIT:2C33h
                                    ; PCDOS 7.1 IBMBIO.COM - SYSINIT:314Eh
                                    ;call   getchr
                                skipline:           ; MSDOS 6.21 IO.SYS - SYSINIT:2C33h
00002CD9 3C0A                       cmp     al,lf  ; 0Ah            ; the safest way to eat the rest of
00002CDB 7405                       je      short endofline             ; the line: watch for ever-present LF
00002CDD E8CF1A                     call    getchr
                                    ; 03/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.1 IO.SYS)
                                    ; (SYSINIT:2C3Ah)
00002CE0 73F7                       jnb     short skipline

                                endofline:
00002CE2 E94AF8                     jmp     conflp

                                    ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                getshparms:
                                    ; 18/12/2022
                                    ; al = 0
00002CE5 8805                       mov     [di],al ; 0
                                    ;mov    byte [di],0             ; zero-terminate the filename
00002CE7 BF[BC4B]                   mov     di,command_line+1       ; prepare to process the command-line
                                parmloop:
00002CEA E8C21A                     call    getchr
00002CED 3C20                       cmp     al," " ; 20h
                                    ;jb     short endofshell
                                    ; 03/01/2023
00002CEF 7209                       jb      short endofparms

00002CF1 8805                       mov     [di],al
00002CF3 47                         inc     di
                                    ;jmp    short parmloop
                                    ; 03/01/2023 - Retro DOS v4.2
```

```
00002CF4 81FF[394C]                 cmp     di,command_line+126
00002CF8 72F0                       jb      short parmloop

                                    ; 03/01/2023 - Retro DOS v4.2
                                endofparms:
00002CFA 89F9                       mov     cx,di
00002CFC 81E9[BC4B]                 sub     cx,command_line+1
                                    ;mov    [cs:command_line],cl
                                    ; 03/01/2023
00002D00 880E[BB4B]                 mov     [command_line],cl
00002D04 EBD0                       jmp     short endofshell

                                ;---------------------------------------------------------------------------
                                ; fcbs command
                                ;---------------------------------------------------------------------------

                                ;****************************************************************************
                                ; function: parse the parameters of fcbs= command.                          *
                                ;                                                                          *
                                ; input :                                                                  *
                                ;   es:si -> parameters in command line.                                   *
                                ; output:                                                                  *
                                ;   set the variables fcbs,keep.                                           *
                                ;                                                                          *
                                ; subroutines to be called:                                                *
                                ;   sysinit_parse                                                          *
                                ; logic:                                                                   *
                                ; {                                                                       *
                                ;   set di points to fcbs_parms;                                           *
                                ;   set dx,cx to 0;                                                        *
                                ;   while (end of command line)                                            *
                                ;   { sysparse;                                                            *
                                ;     if (no error) then                                                   *
                                ;        { if (cx == 1) then /* first positional = fcbs */                 *
                                ;             fcbs = result_val._$P_picked_val;                          *
                                ;          if (cx == 2) then /* second positional = keep */                *
                                ;             keep = result_val._$P_picked_val;                          *
                                ;        }                                                               *
                                ;     else /*error exit*/                                                  *
                                ;        error exit;                                                     *
                                ;   };                                                                     *
                                ; };                                                                      *
                                ;****************************************************************************

                                tryx:
00002D06 80FC58                         cmp     ah,CONFIG_FCBS  ; 'X'
                                    ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
00002D09 7534                       jne     short try1
                                    ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;jne    short tryy      ; comment command

                                ; 31/12/2022 - Retro DOS v4.2
                                ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                ;ifdef      MULTI_CONFIG
00002D0B E80E19                     call    query_user      ; query the user if config_cmd
00002D0E 722F                       jc      short try1      ; has the CONFIG_OPTION_QUERY bit set
                                ;endif
                                ;%endif ; 30/10/2022

00002D10 BF[9E22]                   mov     di,fcbs_parms
00002D13 31C9                       xor     cx,cx
                                    ; 03/01/2023
                                    ;mov    dx,cx
                                do98:
00002D15 E84F01                     call    sysinit_parse
                                        ; 03/01/2023
                                    ;jnc    short if98      ; parse error
                                        ;call      badparm_p      ;  and show messages and end the search loop.
                                    ;jmp    short sr98
                                    ;-----------------------
                                    ; 03/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
00002D18 723C                       jc      short badparm_p_coff
                                if98:
00002D1A 83F8FF                     cmp     ax,_$P_RC_EOL  ; end of line?
00002D1D 7412                       je      short en98      ; then end the $endloop

                                    ;;mov   al,[cs:rv_dword]
                                    ;mov    al,[cs:result_val+_$P_Result_Blk.Picked_Val]
                                    ; 15/12/2022
                                    ; ds = cs
00002D1F A0[1B22]                   mov     al,[result_val+_$P_Result_Blk.Picked_Val]
00002D22 83F901                     cmp     cx,1            ; the first positional?
00002D25 7505                       jne     short if102
                                    ;mov    [cs:p_fcbs],al
                                    ; 15/12/2022
00002D27 A2[D322]                   mov     [p_fcbs],al
                                    ;jmp    short en102
00002D2A EBE9                       jmp     short do98
                                if102:
                                    ;mov    [cs:p_keep],al
                                    ; 15/12/2022
00002D2C A2[D422]                   mov     [p_keep],al
                                en102:
00002D2F EBE4                       jmp     short do98
                                en98:
                                    ; 15/12/2022
                                    ; ds = cs
00002D31 A0[D322]                   mov     al,[p_fcbs]
00002D34 A2[A002]                   mov     [FCBS],al
00002D37 C606[A102]00               mov     byte [KEEP],0
                                    ;mov    al,[cs:p_fcbs]  ; M017
                                    ;mov    [cs:FCBS],al    ; M017
                                    ;mov    byte [cs:KEEP],0 ; M017
                                sr98:
00002D3C E947F8                     jmp     coff

                                ; 31/12/2022 - Retro DOS v4.2
                                %if 0

                                ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;---------------------------------------------------------------------------
                                ; comment= do nothing. just decrease chrptr,and increase count for correct
                                ;          line number
                                ;---------------------------------------------------------------------------

                                tryy:
                                    cmp     ah,CONFIG_COMMENT ; 'Y'
                                    jne     short try0

                                donothing:
                                    ; 15/12/2022
                                    ; ds = cs
                                    dec     word [chrptr]
```

```
                            inc     word [count]
                            ; 02/11/2022
                            ;dec    word [cs:chrptr]
                            ;inc    word [cs:count]

                            jmp     coff

                        ;--------------------------------------------------------------------------
                        ; rem command
                        ;--------------------------------------------------------------------------

                        try0:                           ; do nothing with this line.
                            cmp     ah,CONFIG_REM ; '0'
                            je      short donothing

                        %endif

                        ; 07/04/2019 - Retro DOS v4.0

                        ;--------------------------------------------------------------------------
                        ; switches command
                        ;--------------------------------------------------------------------------

                        ;**************************************************************************
                        ;                                                                        *
                        ; function: parse the option switches specified.                        *
                        ; note - this command is intended for the future use also.                      *
                        ; when we need to set system data flag,use this command.                       *
                        ;                                                                        *
                        ; input :                                                               *
                        ;   es:si -> parameters in command line.                                 *
                        ; output:                                                              *
                        ;   p_swit_k set if /k option chosen.                                  *
                        ;                                                                        *
                        ; subroutines to be called:                                                    *
                        ;   sysinit_parse                                                      *
                        ; logic:                                                               *
                        ; {                                                                     *
                        ;   set di points to swit_parms;  /*parse control definition*/             *
                        ;   set dx,cx to 0;                                                    *
                        ;   while (end of command line)                                        *
                        ;   { sysinit_parse;                                                   *
                        ;     if (no error) then                                               *
                        ;          if (result_val._$P_synonym_ptr == swit_k) then                    *
                        ;               p_swit_k = 1                                         *
                        ;          endif                                                     *
                        ;     else {show error message;error exit}                                 *
                        ;   };                                                                 *
                        ; };                                                                    *
                        ;                                                                        *
                        ;**************************************************************************

                        SUPPRESS_WINA20    EQU 00000010b  ; M025 ; (DOSSYM.INC, MSDOS 6.0)

                        try1:
00002D3F 80FC31                 cmp     ah,CONFIG_SWITCHES ; '1'
00002D42 7403                   je      short do_try1  ; switches= command entered?
                        skip_it5:
                            ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                            ; (SYSINIT:2C8Ah)
00002D44 E98E00             jmp     tryv
                            ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                            ;jmp    tryz

                        do_try1:

                        ; 31/12/2022 - Retro DOS v4.2
                        ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                        ;%if 0
                        ;ifdef      MULTI_CONFIG
00002D47 E8D218             call    query_user      ; query the user if config_cmd
00002D4A 72F8               jc      short skip_it5 ; has the CONFIG_OPTION_QUERY bit set
                        ;endif
                        ;%endif ; 30/10/2022

00002D4C BF[3823]           mov     di,swit_parms
00002D4F 31C9               xor     cx,cx
                            ; 03/01/2023
                            ;mov    dx,cx
                        do110:
00002D51 E81301             call    sysinit_parse
00002D54 7306               jnc     short if110    ; parse error
                            ;call   badparm_p      ;  and show messages and end the search loop.
                            ;jmp    short sr110
                            ; ----------------------
                            ; 03/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                        badparm_p_coff:
00002D56 E83901             call    badparm_p
00002D59 E92AF8             jmp     coff
                            ;-----------------------
                        if110:
00002D5C 83F8FF             cmp     ax,_$P_RC_EOL  ; end of line?
00002D5F 743C               je      short en110    ; then jmp to $endloop for semantic check

                            ; 15/12/2022
                            ; ds = cs
                            ;;cmp   word [cs:result_val_swoff],swit_k
                            ;cmp    word [cs:result_val+_$P_Result_Blk.SYNONYM_Ptr],swit_k
00002D61 813E[1922][5623]   cmp     word [result_val+_$P_Result_Blk.SYNONYM_Ptr],swit_k
00002D67 7507               jne     short if115    ;                               ;M059
                            ; 15/12/2022
00002D69 C606[9523]01       mov     byte [p_swit_k],1
                            ;mov    byte [cs:p_swit_k],1  ; set the flag
00002D6E EBE1               jmp     short do110
                        if115:
                            ; 15/12/2022                                                ;M059
                            ;;cmp   word [cs:result_val_swoff],swit_t
                            ;cmp    word [cs:result_val+_$P_Result_Blk.SYNONYM_Ptr],swit_t     ;M059
00002D70 813E[1922][7A23]   cmp     word [result_val+_$P_Result_Blk.SYNONYM_Ptr],swit_t
                            ;jne    short if116                                 ;M059 M063
                            ; 14/04/2024
                            ;;;
00002D76 7507               jne     short if118  ; (PCDOS 7.1 IBMBIO.COM)
                            ;;;
                            ; 15/12/2022
00002D78 C606[9623]01       mov     byte [p_swit_t],1
                            ;mov    byte [cs:p_swit_t],1                        ;M059
00002D7D EBD2               jmp     short do110                                 ;M059

                            ; 14/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                            ;;;
                        if118:
                            ;;cmp   word [cs:result_val_swoff],swit_i ; offset "/I"
                            ;cmp    word [cs:result_val+_$P_Result_Blk.SYNONYM_Ptr],swit_i
```

```
00002D7F 813E[1922][9223]           cmp     word [result_val+_$P_Result_Blk.SYNONYM_Ptr],swit_i
00002D85 7507                       jne     short if116
                                    ;mov    byte [cs:p_swit_i],1  ; set the flag
00002D87 C606[9823]01               mov     byte [p_swit_i],1
00002D8C EBC3                       jmp     short do110
                                    ;;;
                                if116:
                                    ; 15/12/2022
                                    ;;cmp   word [cs:result_val_swoff],swit_w
                                    ;cmp    word [cs:result_val+_$P_Result_Blk.SYNONYM_Ptr],swit_w     ;M063
00002D8E 813E[1922][8623]           cmp     word [result_val+_$P_Result_Blk.SYNONYM_Ptr],swit_w
00002D94 75BB                       jne     short do110                                ;M063
                                    ; 15/12/2022
00002D96 C606[9723]01               mov     byte [p_swit_w],1
                                    ;mov    byte [cs:p_swit_w],1                       ;M063
00002D9B EBB4                       jmp     short do110                                ;M063
                                en110:
                                    ; 15/12/2022
                                    ; ds = cs
00002D9D 803E[9523]01               cmp     byte [p_swit_k],1
                                    ;cmp    byte [cs:p_swit_k],1  ; if /k entered,
00002DA2 1E                         push    ds
                                    ;;mov   ax,Bios_Data
                                    ;mov    ax,KERNEL_SEGMENT ; 0070h
                                    ; 21/10/2022
00002DA3 B87000                     mov     ax,DOSBIODATASEG ; 0070h
00002DA6 8ED8                       mov     ds,ax
00002DA8 750A                       jne     short if117
                                    ; 14/04/2024
00002DAA C606[7E04]00               mov     byte [keyrd_func],0 ; 4E5h ; use the conventional keyboard functions
                                                                ; BIOSDATA:047Eh for PCDOS 7.1 IBMBIO.COM
00002DAF C606[7F04]01               mov     byte [keysts_func],1 ; 4E6h (for MSDOS 6.21 IO.SYS)
                                                                ; BIOSDATA:047Fh for PCDOS 7.1 IBMBIO.COM
                                if117:
                                    ; 15/12/2022
                                    ; ds <> cs
00002DB4 2EA0[9623]                 mov     al,[cs:p_swit_t]                           ;M059
00002DB8 A2[8B04]                   mov     [t_switch],al  ; 4F2h (for MSDOS 6.21 IO.SYS)      ;M059
                                                        ; 14/04/2024   ; BIOSDATA:048Bh for PCDOS 7.1 IBMBIO.COM
00002DBB 2E803E[9723]00             cmp     byte [cs:p_swit_w],0                       ;M063
00002DC1 740E                       je      short skip_dos_flag                        ;M063
00002DC3 06                         push    es
00002DC4 53                         push    bx
00002DC5 B452                       mov     ah,GET_IN_VARS ; 52h                       ;M063
00002DC7 CD21                       int     21h                                        ;M063
                                                ; DOS - 2+ internal - GET LIST OF LISTS
                                                ; Return: ES:BX -> DOS list of lists
                                    ;or     bytes [es:86h],2
00002DC9 26800E860002               or      byte [es:DOS_FLAG_OFFSET],SUPPRESS_WINA20 ; 2      ;M063
00002DCF 5B                         pop     bx
00002DD0 07                         pop     es
                                skip_dos_flag:                                         ;M063
00002DD1 1F                         pop     ds
                                sr110:
00002DD2 E9B1F7                     jmp     coff

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:2D14h)
                                ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0

                                tryv:

                                ;ifdef      MULTI_CONFIG
                                ;--------------------------------------------------------------------------
                                ; set command (as in "set var=value<cr/lf>")
                                ;--------------------------------------------------------------------------

00002DD5 80FC56                     cmp     ah,CONFIG_SET  ; 'V'
00002DD8 750F                       jne     short tryn
00002DDA E83F18                     call    query_user      ; query the user if config_cmd
00002DDD 720A                       jc      short tryn     ; has the CONFIG_OPTION_QUERY bit set
00002DDF E83614                     call    copy_envvar     ; copy var at ES:SI to "config_wrkseg"
00002DE2 73EE                       jnc     short sr110    ; no error
                                err:
00002DE4 E8D300                     call    error_line      ; whoops, display error in line XXX
00002DE7 EBE9                       jmp     short sr110     ; jump to coff (to skip to next line)

                                ;--------------------------------------------------------------------------
                                ; numlock command (as in "numlock=on|off")
                                ;--------------------------------------------------------------------------
                                tryn:
00002DE9 80FC4E                     cmp     ah,CONFIG_NUMLOCK  ;'N'
                                    ;jne    short tryy
                                    ; 14/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
00002DEC 750C                       jne     short tryt

00002DEE E82B18                     call    query_user      ; query the user if config_cmd
00002DF1 7238                       jc      short tryy     ; has the CONFIG_OPTION_QUERY bit set
00002DF3 E8B710                     call    set_numlock
00002DF6 72EC                       jc      short err
00002DF8 EBD8                       jmp     short sr110    ; all done

                                ;endif      ;MULTI_CONFIG

                                ; 14/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
                                ;--------------------------------------------------------------------------
                                ; dosdata command
                                ;--------------------------------------------------------------------------
                                tryt:
                                    ;cmp    ah,54h ; 'T'
00002DFA 80FC54                     cmp     ah,CONFIG_DOSDATA ; 'T' ; PCDOS 7 new config cmd
00002DFD 752C                       jne     short tryy

00002DFF E81A18                     call    query_user
00002E02 7227                       jc      short tryy

00002E04 BF[C823]                   mov     di,dosdata_parms
00002E07 31C9                       xor     cx,cx
                                    ; 14/04/2024 - Retro DOS v5.0
                                    ;mov    dx,cx ; 0
                                do120:
00002E09 E85B00                     call    sysinit_parse
00002E0C 7303                       jnc     short if120

                                    ;call   badparm_p
                                    ;jmp    short en120
                                    ; 14/04/2024 - Retro DOS v5.0
00002E0E E945FF                     jmp     badparm_p_coff
                                if120:
                                    ;cmp    ax,0FFFFh
00002E11 83F8FF                     cmp     ax,_$P_RC_EOL ; -1 ; end of line?
00002E14 7422                       jz      short en120
00002E16 803E[1822]01               cmp     byte [result_val_itag],1 ; tag 1 (UMB)
```

```
                                                    ; [result_val+_$P_Result_Blk.Item_Tag]
00002E1B 7507                       jnz     short if121
00002E1D C606[6E03]01               mov     byte [dosdata_umb],1 ; DOSDATA=UMB (1) NOUMB (0)
                                    ;jmp    short sr120
                                    ; 14/04/2024
00002E22 EBE5                       jmp     short do120
                                if121:
00002E24 C606[6E03]00               mov     byte [dosdata_umb],0 ; DOSDATA=UMB (1) NOUMB (0)
                                sr120:
00002E29 EBDE                       jmp     short do120
                                    ; 14/04/2024
                                ;en120:
                                    ;jmp    coff

                                ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;--------------------------------------------------------------------------
                                ; comment= do nothing. just decrease chrptr,and increase count for correct
                                ;          line number
                                ;--------------------------------------------------------------------------

                                    ; 31/12/2022
                                tryy:
00002E2B 80FC59                     cmp     ah,CONFIG_COMMENT ; 'Y'
00002E2E 750B                       jne     short try0

                                donothing:
                                    ; 15/12/2022
                                    ; ds = cs
00002E30 FF0E[5A03]                 dec     word [chrptr]
00002E34 FF06[5603]                 inc     word [count]
                                    ; 02/11/2022
                                    ;dec    word [cs:chrptr]
                                    ;inc    word [cs:count]
                                en120:              ; 14/04/2024
00002E38 E94BF7                     jmp     coff

                                ;--------------------------------------------------------------------------
                                ; rem command
                                ;--------------------------------------------------------------------------

                                try0:                           ; do nothing with this line.
00002E3B 80FC30                     cmp     ah,CONFIG_REM ; '0'
00002E3E 74F0                       je      short donothing

                                ;%endif     ; 30/10/2022

                                ; 30/10/2022
                                ; (MSSOS 5.0 IO.SYS - SYSINIT:29D7h)

                                ;--------------------------------------------------------------------------
                                ; bogus command
                                ;--------------------------------------------------------------------------

                                tryz:
00002E40 80FCFF                         cmp     ah,0FFh           ;null command? (BUGBUG - who sets FFh anyway?)
                                    ; 31/12/2022
00002E43 74EB                       je      short donothing
                                    ; 02/11/2022
                                    ;je     short tryz_donothing

00002E45 FF0E[5A03]                 dec     word [chrptr]
00002E49 FF06[5603]                 inc     word [count]
00002E4D EB37                       jmp     short badop

                                ; 31/12/2022
                                ;   ; 02/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ;tryz_donothing:
                                ;   jmp     donothing

                                ; 07/04/2019 - Retro DOS v4.0

                                ;--------------------------------------------------------------------------

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:2D5Dh)

                                ; 11/07/2023 - Retro DOS v4.2 IO.SYS (optimization)

                                ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;
                                ;***CheckProtmanArena -- special hack for adjusting alloclim with Protman$
                                ;
                                ;   adjusts alloclim if Protman$ reduced our arena through a manual hack.
                                ;
                                CheckProtmanArena:
                                    ; 08/09/2023
                                    ; ds = cs
00002E4F 06                         push    es
                                    ;mov    ax,[cs:area]   ; get our arena header
00002E50 A1[6803]                   mov     ax,[area] ; 08/09/2023
00002E53 48                         dec     ax
00002E54 8EC0                       mov     es,ax
                                    ;add    ax,[es:ARENA.SIZE]
00002E56 2603060300                 add     ax,[es:3]       ; find end of arena
00002E5B 40                         inc     ax
                                    ; 08/09/2023
00002E5C 3B06[A502]                 cmp     ax,[ALLOCLIM]
                                    ;cmp    ax,[cs:ALLOCLIM] ; is it less than alloclim?
00002E60 7703                       ja      short CheckProtmanDone

                                    ;mov    [cs:ALLOCLIM],ax ; reduce alloclim then
                                    ; 08/09/2023
00002E62 A3[A502]                   mov     [ALLOCLIM],ax
                                CheckProtmanDone:
00002E65 07                         pop     es
00002E66 C3                         retn

                                ;--------------------------------------------------------------------------

                                sysinit_parse:

                                ;--------------------------------------------------------------------------
                                ;set up registers for sysparse
                                ;in)es:si -> command line in confbot
                                ;   di -> offset of the parse control definition.
                                ;
                                ;out)       calls sysparse.
                                ;   carry will set if parse error.
                                ;   *** the caller should check the eol condition by looking at ax
                                ;   *** after each call.
                                ;   *** if no parameters are found,then ax will contain a error code.
                                ;   *** if the caller needs to look at the synomym@ of the result,
                                ;   ***  the caller should use cs:@ instead of es:@.
                                ;   cx register should be set to 0 at the first time the caller calls this
```

```
                                ;    procedure.
                                ;   ax - exit code
                                ;   bl - terminated delimeter code
                                ;   cx - new positional ordinal
                                ;   si - set to pase scanned operand
                                ;   dx - selected result buffer
                                ;--------------------------------------------------------------------------

                                    ; 03/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:2D78h)

                                    ; 14/04/2024 - Retro DOS v5.0
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:32F3h)

                                    ; 09/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
                                    ; ds = cs
00002E67 8C06[6D19]                 mov     [badparm_seg],es      ;save the pointer to the parm
00002E6B 8936[6B19]                 mov     [badparm_off],si      ;we are about to parse for badparm msg.

                                    ; 24/10/2022
00002E6F 06                         push    es                    ;save es,ds
00002E70 1E                         push    ds

00002E71 06                         push    es
00002E72 1F                         pop     ds                    ;now ds:si -> command line

00002E73 0E                         push    cs
00002E74 07                         pop     es                    ;now es:di -> control definition

                                    ; 09/09/2023
                                    ;mov    [cs:badparm_seg],ds   ;save the pointer to the parm
                                    ;mov    [cs:badparm_off],si   ;we are about to parse for badparm msg.

                                    ;mov    dx,0
                                    ; 04/01/2023
00002E75 29D2                       sub     dx,dx ; 0
00002E77 E89BEB                     call    SysParse
                                    ;cmp    ax,_$P_No_Error       ; 0     ;no error
                                    ; 06/09/2023
00002E7A 21C0                       and     ax,ax

                                ;**cas note: when zero true after cmp, carry clear

                                    ;je     short ll4
                                    ; 24/10/2022 (MSDOS 5.0 IO.SYS compatibility, SYSINIT:2A02h)
                                    ; 12/12/2022
00002E7C 7405                       je      short en4 ; cf=0
00002E7E 83F8FF                     cmp     ax,_$P_RC_EOL ; 0FFFFh;or the end of line?
                                    ;jne    short if4
                                    ; 12/12/2022
00002E81 7400                       je      short en4 ; cf=0
                                    ; 06/09/2023
                                    ; cf=1

                                ; 12/12/2022
                                ;ll4:
                                ;   ; 12/12/2022
                                ;   ; cf=0
                                ;   ;clc
                                ;   jmp     short en4

                                if4:
                                    ; 24/10/2022
                                    ; 06/09/2023 (cf=1)
                                    ;stc
                                en4:
00002E83 1F                         pop     ds
00002E84 07                         pop     es
00002E85 C3                         retn

                                ; 11/12/2022
                                %if 0

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : badop_p
                                ;
                                ;             same thing as badop,but will make sure to set ds register back
                                ;             to sysinitseg and return back to the caller.
                                ;
                                ;--------------------------------------------------------------------------

                                badop_p:
                                    push    cs
                                    pop     ds              ;set ds to configsys seg.
                                    mov     dx,badopm
                                    call    print
                                        ;call      error_line
                                    ;retn
                                    ; 11/12/2022
                                    jmp     error_line

                                %endif

                                ;--------------------------------------------------------------------------
                                ;
                                ; label : badop
                                ;
                                ;--------------------------------------------------------------------------

                                badop:
00002E86 BA[4C50]                   mov     dx,badopm       ;want to print command error "unrecognized command..."
00002E89 E8C81B                     call    print
00002E8C E82B00                     call    error_line      ;show "error in config.sys ..." .
00002E8F E9F4F6                     jmp     coff

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : badparm_p
                                ;
                                ;             show "bad command or parameters - xxxxxx"
                                ;             in badparm_seg,badparm_off -> xxxxx
                                ;
                                ;--------------------------------------------------------------------------

                                    ; 24/10/2022
                                badparm_p:
                                    ; 11/12/2022
                                    ; ds = cs
                                    ; 11/12/2022
                                    ;push   ds ; *
00002E92 52                         push    dx
00002E93 56                         push    si

```

```
                                    ; 11/12/2022
                                    ; ds = cs
                                    ;push   cs
                                    ;pop    ds

00002E94 BA[7350]                   mov     dx,badparm
00002E97 E8BA1B                     call    print                   ; "bad command or parameters - "
00002E9A C536[6B19]                 lds     si,[badparm_ptr]

                                ;   print "xxxx" until cr.

                                do1:
00002E9E 8A14                       mov     dl,[si]                 ; get next character
00002EA0 80FA0D                     cmp     dl,cr ; 0Dh             ; is a carriage return?
00002EA3 7407                       je      short en1               ; exit loop if so

00002EA5 B402                       mov     ah,2 ; STD_CON_OUTPUT ; function 2
00002EA7 CD21                       int     21h                     ; display character
00002EA9 46                         inc     si                      ; next character
00002EAA EBF2                       jmp     short do1
                                en1:
00002EAC 0E                         push    cs
00002EAD 1F                         pop     ds

00002EAE BA[7050]                   mov     dx,crlfm
00002EB1 E8A01B                     call    print
00002EB4 E80300                     call    error_line

00002EB7 5E                         pop     si
00002EB8 5A                         pop     dx
                                    ; 11/12/2022
                                    ;pop    ds ; *
                                badparmp_ret:
00002EB9 C3                         retn

                                ; 10/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
                                %if 0

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : getchr
                                ;
                                ;--------------------------------------------------------------------------

                                    ; 24/10/2022
                                getchr:
                                    ; 12/12/2022
                                    ;push   cx
                                    ;mov    cx,[count]
                                    ;jcxz   nochar
                                    ; 12/12/2022
                                    cmp     word [count],1
                                    jb      short nochar ; cf=1 ([count] = 0)

                                    mov     si,[chrptr]
                                    mov     al,[es:si]
                                    dec     word [count]
                                    inc     word [chrptr]
                                    ; 12/12/202
                                    ; cf=0
                                    ;clc
                                ;get_ret:
                                    ;pop    cx
                                    ;retn
                                nochar:
                                    ; 12/12/2022
                                    ; cf=1
                                    ;stc
                                    ;jmp    short get_ret

                                    retn
                                %endif

                                ; 11/12/2022
                                %if 0

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : incorrect_order
                                ;
                                ;             show "incorrect order in config.sys ..." message.
                                ;
                                ;--------------------------------------------------------------------------

                                incorrect_order:
                                    mov     dx,badorder
                                    call    print
                                    call    showlinenum
                                    retn

                                %endif

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : error_line
                                ;
                                ;             show "error in config.sys ..." message.
                                ;
                                ;--------------------------------------------------------------------------

                                    ; 11/12/2022
                                    ; 24/10/2022
                                error_line:
                                    ; 11/12/2022
                                    ; ds = cs
                                    ;push   cs
                                    ;pop    ds

00002EBA BA[A851]                   mov     dx,errorcmd
00002EBD E8941B                     call    print
                                    ;call   showlinenum
                                    ;retn
                                    ; 11/12/2022
                                    ;jmp    short shortlinemum

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : showlinenum
                                ;
                                ; convert the binary linecount to decimal ascii string in showcount
                                ; and display showcount at the current curser position.
                                ; in.) linecount
                                ;
                                ; out) the number is printed.
```

```
                                ;
                                ;--------------------------------------------------------------------------
                                
                                    ; 11/12/2022
                                    ; ds = cs
                                    ; 24/10/2022
                                showlinenum:
00002EC0 06                         push    es
                                    ; 11/12/2022
                                    ;push   ds
00002EC1 57                         push    di
                                
00002EC2 0E                         push    cs
00002EC3 07                         pop     es              ; es=cs
                                
                                    ; 11/12/2022
                                    ;push   cs
                                    ;pop    ds
                                
00002EC4 BF[B502]                   mov     di,showcount+4 ; di -> the least significant decimal field.
00002EC7 B90A00                     mov     cx,10          ; decimal divide factor
                                    ;mov    ax,[cs:linecount]
                                    ; 11/12/2022
00002ECA A1[AF02]                   mov     ax,[linecount]
                                sln_loop:
                                    ; 11/12/2022
00002ECD 39C8                       cmp     ax,cx ; < 10 ?
                                    ;cmp    ax,10          ; < 10?
00002ECF 720C                       jb      short sln_last
                                
00002ED1 31D2                       xor     dx,dx
00002ED3 F7F1                       div     cx      ; cx = 10
00002ED5 80CA30                     or      dl,30h         ; add "0" (= 30h) to make it an ascii.
00002ED8 8815                       mov     [di],dl
00002EDA 4F                         dec     di
00002EDB EBF0                       jmp     short sln_loop
                                
                                sln_last:
00002EDD 0C30                       or      al,30h ; "0"
00002EDF 8805                       mov     [di],al
00002EE1 89FA                       mov     dx,di
00002EE3 E86E1B                     call    print          ; show it.
00002EE6 5F                         pop     di
                                    ; 11/12/2022
                                    ;pop    ds
00002EE7 07                         pop     es
00002EE8 C3                         retn
                                
                                ; 07/04/2019 - Retro DOS v4.0
                                ; (MSDOS 6.21 IO.SYS, SYSINIT:2E44h)
                                
                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : ProcDOS
                                ;
                                ;   Process the result of DOS= parsing
                                ;
                                ;   result_val._$P_item_tag       = 1 for DOS=HIGH
                                ;                           = 2 for DOS=LOW
                                ;                           = 3 for DOS=UMB
                                ;                           = 4 for DOS=NOUMB
                                ;--------------------------------------------------------------------------
                                
                                    ; 01/11/2022 - Retro DOS v4.0 (Modififed MSDOS 5.0 IO.SYS)
                                    ; (SYTSINIT:2AB5h)
                                ProcDOS:
                                    ; 01/01/2023
                                    ; ds = cs
00002EE9 30E4                       xor     ah,ah
                                    ;;mov   al,[cs:result_val_itag]
                                    ;mov    al,[cs:result_val+_$P_Result_Blk.Item_Tag]
                                    ; 01/01/2023
00002EEB A0[1822]                   mov     al,[result_val+_$P_Result_Blk.Item_Tag]
00002EEE 48                         dec     ax
00002EEF 7415                       jz      short pd_hi
00002EF1 48                         dec     ax
00002EF2 740E                       jz      short pd_lo
00002EF4 48                         dec     ax
00002EF5 7405                       jz      short pd_umb
                                    ;;mov   byte [cs:DevUMB],0
                                    ; 18/12/2022
                                    ;mov    byte [cs:DevUMB],ah ; 0
                                    ; 01/01/2023
00002EF7 8826[4224]                 mov     byte [DevUMB],ah ; 0
00002EFB C3                         retn
                                pd_umb:
                                    ; 01/01/2023
00002EFC C606[4224]FF               mov     byte [DevUMB],0FFh
                                    ;mov    byte [cs:DevUMB],0FFh
00002F01 C3                         retn
                                pd_lo:
                                    ; 01/01/2023
00002F02 A2[6C02]                   mov     [runhigh],al ; 0
                                    ; 18/12/2022
                                    ;mov    [cs:runhigh],al ; 0
                                    ;;mov   byte [cs:runhigh],0
00002F05 C3                         retn
                                pd_hi:
                                    ; 01/01/2023
00002F06 C606[6C02]FF               mov     byte [runhigh],0FFh
                                    ;mov    byte [cs:runhigh],0FFh
                                limx:       ; 11/12/2022
00002F0B C3                         retn
                                
                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : LieInt12Mem
                                ;
                                ;   Input : DevEntry points to Device Start address (offset == 0)
                                ;           alloclim set to the limit of low memory.
                                ;
                                ;   Output : none
                                ;
                                ;   Changes the ROM BIOS variable which stores the total low memory
                                ;   If a 3com device driver (any character device with name 'PROTMAN$')
                                ;   is being loaded alloclim is converted into Ks and stored in 40:13h
                                ;   Else if a device driver being loaded into UMB the DevLoadEnd is
                                ;   converted into Ks and stored in 40:13h
                                ;
                                ;--------------------------------------------------------------------------
                                
                                LieInt12Mem:
                                    ; 11/12/2022
                                    ; ds = cs
```

```
00002F0C A1[A502]                   mov     ax,[ALLOCLIM]
                                    ;mov    ax,[cs:ALLOCLIM]      ; lie INT 12 as alloclim
                                                                  ; assuming that it is 3Com
00002F0F E84200                     call    IsIt3Com              ; Is it 3Com driver?
00002F12 740A                       jz      short lim_set         ; yes, lie to him differently
                                    ; 13/05/2019
                                    ;cmp    byte [cs:DeviceHi],0  ; Is the DD being loaded in UMB
                                    ;je     short limx            ; no, don't lie
                                    ;mov    ax,[cs:DevLoadEnd]    ; lie INT 12 as end of UMB
                                    ; 11/12/2022
                                    ; ds = cs
00002F14 803E[5124]00               cmp     byte [DeviceHi],0
00002F19 74F0                       je      short limx
00002F1B A1[3724]                   mov     ax,[DevLoadEnd]
                                lim_set:
                                    ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ; 11/12/2022
                                    ;call   SetInt12Mem
                                ;limx:
                                    ;retn

                                    ;jmp    short SetInt12Mem

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : SetInt12Mem
                                ;
                                ;   Input : AX = Memory size to be set (in paras)
                                ;   Output : none
                                ;
                                ;   Sets the variable 40:13 to the memory size passed in AX
                                ;   It saves the old value in 40:13 in OldInt12Mem,
                                ;   It also sets a flag Int12Lied to 0ffh, which is checked before
                                ;   restoring the value of 40:13
                                ;
                                ;--------------------------------------------------------------------------

                                    ; 01/11/2022
                                SetInt12Mem:
00002F1E 1E                         push    ds
00002F1F BB4000                     mov     bx,40h
00002F22 8EDB                       mov     ds,bx                 ; ROM BIOS Data Segment
00002F24 8B1E1300                   mov     bx,[13h]              ; INT 12 memory variable
                                    ;mov    [cs:OldInt12Mem],bx   ; save it
00002F28 B106                       mov     cl,6
00002F2A D3E8                       shr     ax,cl                 ; convert paras into Ks
00002F2C A31300                     mov     [13h],ax              ; Lie
                                    ;mov    byte [cs:Int12Lied],0FFh ; mark that we are lying
00002F2F 1F                         pop     ds
                                    ; 14/04/2024
                                    ; ds = cs
00002F30 891E[5524]                 mov     [OldInt12Mem],bx
00002F34 C606[5424]FF               mov     byte [Int12Lied],0FFh
                                ;limx:
00002F39 C3                         retn

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : TrueInt12Mem
                                ;
                                ;   Input : Int12Lied = 0 if we are not lying currently
                                ;                     = 0ffh if we are lying
                                ;           OldInt12Mem = Saved value of 40:13h
                                ;
                                ;   Output : none
                                ;
                                ;   Resets the INT 12 Memory variable if we were lying about int 12
                                ;   and resets the flag which indicates that we were lying
                                ;
                                ;--------------------------------------------------------------------------

                                TrueInt12Mem:
                                    ; 11/12/2022
                                    ; ds = cs
00002F3A 803E[5424]00               cmp     byte [Int12Lied],0
                                    ;cmp    byte [cs:Int12Lied],0 ; were we lying so far?
                                    ; 01/11/2022 (MSDOS 5.0 IO.SYS, SYS.INIT:2B1Dh)
                                    ;mov    byte [cs:Int12Lied],0 ; reset it anyway
00002F3F 7412                       je      short timx            ; no, we weren't
                                    ; 18/12/2022
00002F41 B84000                     mov     ax,40h
00002F44 8826[5424]                 mov     [Int12Lied],ah ; 0
                                    ;mov    byte [Int12Lied],0
                                    ;mov    byte [cs:Int12Lied],0
00002F48 1E                         push    ds
                                    ;mov    ax,40h
00002F49 8ED8                       mov     ds,ax
00002F4B 2EA1[5524]                 mov     ax,[cs:OldInt12Mem]
00002F4F A31300                     mov     [13h],ax              ; restore INT 12 memory
00002F52 1F                         pop     ds
                                timx:
00002F53 C3                         retn

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : IsIt3Com?
                                ;
                                ;   Input : DevEntry = Seg:0 of device driver
                                ;   Output : Zero flag set if device name is 'PROTMAN$'
                                ;            else Zero flag is reset
                                ;
                                ;--------------------------------------------------------------------------

                                IsIt3Com:
                                    ; 11/12/2022
                                    ; ds = cs
00002F54 1E                         push    ds
00002F55 06                         push    es
00002F56 56                         push    si
                                    ; 11/12/2022
00002F57 C536[3924]                 lds     si,[DevEntry]
                                    ;lds    si,[cs:DevEntry]      ; ptr to device header
00002F5B 83C60A                     add     si,SYSDEV.NAME ; 10   ; ptr device name
00002F5E 0E                         push    cs
00002F5F 07                         pop     es
00002F60 BF[5724]                   mov     di,ThreeComName
00002F63 B90800                     mov     cx,8                  ; name length
00002F66 F3A6                       rep     cmpsb
00002F68 5E                         pop     si
00002F69 07                         pop     es
00002F6A 1F                         pop     ds
00002F6B C3                         retn

                                ;M020 : BEGIN
```

```
                                ;--------------------------------------------------------------------------

                                UpdatePDB:
00002F6C 1E                         push    ds
00002F6D B462                       mov     ah,62h
00002F6F CD21                       int     21h     ; DOS - 3+ - GET PSP ADDRESS
00002F71 8EDB                       mov     ds,bx
00002F73 2E8B1E[A502]               mov     bx,[cs:ALLOCLIM]
                                    ;mov    [2],bx
00002F78 891E0200                   mov     [PDB.BLOCK_LEN],bx
00002F7C 1F                         pop     ds
00002F7D C3                         retn

                                ; M020 : END

                                ;--------------------------------------------------------------------------

                                ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:2EEEh)

                                ;include highload.inc     ; Routines for devicehigh parsing, control of HIDDEN
                                ;include highexit.inc     ; umb's, etc

                                ; ----------------------------------------------------------------------
                                ; HIGHLOAD.INC (MSDOS 6.0 - 1991)
                                ; ----------------------------------------------------------------------
                                ; 07/04/2019 - Retro DOS v4.0

                                ;**************************************************************************
                                ;
                                ; This file contains routines needed to parse and implement user-given
                                ; command-line options of the form "/S/L:3,0x500;2;7,127;0x0BE4". InitVar()
                                ; and Parsevar() are used to parse this data and place it in encoded form into
                                ; the variables in highvar.inc, for use by the rest of the routines.
                                ;
                                ; DeviceHigh accepts this command-line (handled in sysconf.asm, not here):
                                ;    DEVICEHIGH SIZE=hhhhhh module opts
                                ; Or, DeviceHigh and LoadHigh accept any of the following:
                                ;    DH/LH module opts
                                ;    DH/LH [/S][/L:umb[,size][;umb[,size]]*] module opts
                                ;    DH/LH [/L:umb[,size][;umb[,size]]*][/S] module opts
                                ; The initial UMB,SIZE pair designates the module's load address; the remainder
                                ; of the UMB and SIZE pairs are used to indicate specific UMBs to be left
                                ; available during the load.
                                ;
                                ; When an actual load is ready to be performed, a call to HideUMBs() will
                                ; temporarily allocate (as owner 8+"HIDDEN  ") all free elements in any
                                ; upper-memory block which was not specified by the user... in addition, if
                                ; UMBs were marked to shrink (/S option) to a certain size ("umb,size"), any
                                ; elements in that umb SAVE the lower-half of the newly-shrunken one are also
                                ; allocated. After the load, the function UnHideUMBs() (in highexit.inc) will
                                ; free any UMBs so allocated.
                                ;
                                ; When a device driver loads, there is the additional problem of allocating its
                                ; initial load site; this should be restricted to the first UMB specified on
                                ; the command-line. The function FreezeUM temporarily allocates all remaining
                                ; free upper-memory elements (as owner 8+"FROZEN  "), except those in the load
                                ; UMB. Then the initial allocation may be made, and a call to UnFreeze will
                                ; return any so-allocated memory elements to FREE, for the true load. Note
                                ; that UnFreeze leaves HIDDEN elements allocated; it only frees FROZEN ones.
                                ;
                                ;**************************************************************************

                                SWTCH       equ     '/'             ; Switch character

                                DOS_CHECK_STRATEGY  equ    5800h   ; Int 21h, Func 58h, Svc 0 = check alloc strat
                                DOS_SET_STRATEGY    equ    5801h   ; Int 21h, Func 58h, Svc 1 = set alloc strategy
                                DOS_CHECK_UMBLINK   equ    5802h   ; Int 21h, Func 58h, Svc 2 = check link state
                                DOS_GET_UMBLINK        equ 5802h ; 20/04/2019
                                DOS_SET_UMBLINK     equ    5803h   ; Int 21h, Func 58h, Svc 3 = set link state
                                DOS_GET_DOS_LISTS   equ      52h   ; Int 21h, Func 52h = return list of lists
                                DOS_UMB_HEAD        equ      8Ch   ; Offset from ES (after func52h) to get UMBHead

                                CR  equ     0Dh             ; Carriage Return
                                LF  equ     0Ah             ; Line Feed
                                TAB equ     09h             ; Tab character (^I)

                                ; ----------------------------------------------------------------------------
                                ;*** InitVar - initializes all the variables used in ParseVar and HideUMBs
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:       None
                                ; EXIT:        Variables listed in highvar.inc are initialized
                                ; ERROR EXIT:  None
                                ; USES:        Flags, variables in highvar.inc
                                ; ----------------------------------------------------------------------------
                                ; Note that element 0 references UMB 0 (conventional), not UMB 1. Its contents
                                ; are largely ignored, but it is initialized nonetheless.
                                ; ----------------------------------------------------------------------------

                                ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:2EEEh)

                                InitVar:
                                    ; 01/01/2023
                                    ; ds = cs

                                    ;pushreg <ax, cx, di, es>
                                    ; 03/01/2023
                                    ;push  ax
                                    ;push  cx
                                    ;push  di
00002F7E 06                         push    es

                                    ;dataseg es                 ;Point ES into appropriate data segment
00002F7F 0E                         push    cs
00002F80 07                         pop     es

00002F81 31C0                       xor     ax,ax
                                    ;mov    [es:fUmbTiny],al    ;Shrink UMBs? (made 1 if /S given)
                                    ;mov    [es:fInHigh],al             ;Set to 1 when DH/LH has been called
                                    ;mov    [es:SegLoad],ax             ;Load Address (seg), used for DH only
                                    ;mov    byte [es:UmbLoad],UNSPECIFIED ; 0FFh
                                    ;                           ;Later is the # of the 1st spec'd UMB
                                    ;mov    [es:fm_argc], al    ;Start with zero args having been read

                                    ; 01/01/2023
                                    ; ds = cs
00002F83 A2[FC23]                   mov     [fUmbTiny],al       ;Shrink UMBs? (made 1 if /S given)
00002F86 A2[FB23]                   mov     [fInHigh],al        ;Set to 1 when DH/LH has been called
00002F89 A3[FD23]                   mov     [SegLoad],ax        ;Load Address (seg), used for DH only
00002F8C C606[FF23]FF               mov     byte [UmbLoad],UNSPECIFIED ; 0FFh
```

```
                                                        ;Later is the # of the 1st spec'd UMB
00002F91 A2[3224]               mov     [fm_argc], al   ;Start with zero args having been read

00002F94 FC                     cld

00002F95 B91000                 mov     cx,MAXUMB ; 16  ;For each entry
00002F98 BF[0024]               mov     di,UmbUsed      ;on the UmbUsed array,
00002F9B F3AA                   rep     stosb           ;      Store 0

                                ;mov    cx,MAXUMB ; 16  ;Okay... for each entry
                                ; 01/01/2033
00002F9D B110                   mov     cl,MAXUMB ; 16
00002F9F BF[1024]               mov     di,UmbSize      ;on the UmbSize array,
00002FA2 F3AB                   rep     stosw           ;      Store 0

                                ;normseg es             ; Return ES

                                ;popreg <es, di, cx, ax>
00002FA4 07                     pop     es
                                ; 03/01/2023
                                ;pop    di
                                ;pop    cx
                                ;pop    ax

00002FA5 C3                     retn

                        ; ---------------------------------------------------------------------------
                        ;*** FixMem - scans the upper memory chain and concatenates adjacent free MCBs
                        ; ---------------------------------------------------------------------------
                        ; ENTRY   : None
                        ; EXIT    : None
                        ; ERROR   : None
                        ; USES    : Flags, fm_umb, fm_strat
                        ; ---------------------------------------------------------------------------

                            ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                            ; (SYSINIT:2F22h)
                        FixMem:
                            ; 01/01/2023
                            ;push   ax
                            ;push   bx
                            ;push   cx
                            ;push   dx
00002FA6 06                 push    es

00002FA7 E84900             call    fm_link         ; Link in UMBs

00002FAA E80002             call    UmbHead         ; Get first upper-memory MCB address (0x9FFF)
00002FAD 723F               jc      short fmX       ; (if couldn't get it, leave now).

00002FAF 8EC0               mov     es,ax           ; It returns in AX, so move it to ES.

                        ; - Walk MCB Chain ------------------------------------------------------------

00002FB1 31D2               xor     dx,dx           ; We're keeping the address of the last MCB
00002FB3 89D1               mov     cx,dx           ; in CX... and the last owner
00002FB5 42                 inc     dx              ; in dx as we go through the loop:

                        ; ----------------------------------------
                        ; FM10--DX  = last MCB's owner's PSP address
                        ;       CX  = last MCB's address (segment)
                        ; ----------------------------------------

00002FB6 26A00000       fm10:       mov     al,[es:ARENA.SIGNATURE] ; if 'Z', don't repeat loop
00002FBA 268B1E0100             mov     bx,[es:ARENA.OWNER]     ; if not zero, do nothing
00002FBF 09D3                   or      bx,dx                   ; dx was owner of previous MCB
00002FC1 7516                   jnz     short fm30              ; If not both zero, don't cat.

                                ; - Coalesce memory blocks at ES:00 and CX:00 -------------------------

00002FC3 268B1E0300     fm20:       mov     bx,[es:ARENA.SIZE]      ; Grab this block's Size,
00002FC8 8EC1                   mov     es,cx                   ; Go back to prev MCB's address
00002FCA 26A20000               mov     [es:ARENA.SIGNATURE],al ; & move the SECOND sig here

00002FCE 26031E0300             add     bx,[es:ARENA.SIZE]      ; Size += first MCB's size
                                ;add    bx,1                    ; And add one for the header
                                ; 11/07/2023
00002FD3 43                     inc     bx
00002FD4 26891E0300             mov     [es:ARENA.SIZE],bx      ; Write the size

                                ; ---------------------------------------------------------------------

00002FD9 8CC1           fm30:       mov     cx,es                   ; Put this address on the stack
00002FDB 268B160100             mov     dx,[es:ARENA.OWNER]     ; And remember its owner

00002FE0 8CC3                   mov     bx,es                   ; Move to the next MCB
00002FE2 26031E0300             add     bx,[es:ARENA.SIZE]
00002FE7 43                     inc     bx
00002FE8 8EC3                   mov     es,bx

                                ;cmp    al,'Z'
00002FEA 3C5A                   cmp     al,arena_signature_end
00002FEC 75C8                   jne     short fm10              ; If signature != 'Z', there are more.
                        fmX:
00002FEE E81300                 call    fm_unlink               ; Unlink UMBs

00002FF1 07                     pop     es
                                ; 01/01/2023
                                ;pop    dx
                                ;pop    cx
                                ;pop    bx
                                ;pop    ax

00002FF2 C3                     retn

                        ; ---------------------------------------------------------------------------
                        ;*** fm_link - links UMBs not already linked in
                        ; ---------------------------------------------------------------------------
                        ; ENTRY:    None
                        ; EXIT:     fm_umb == 0 if not linked in previously, 1 if already linked in
                        ; ERROR:    None
                        ; USES:     AX, BX, fm_umb
                        ; ---------------------------------------------------------------------------

                            ; 01/01/2023 - Retro DOS v4.2
                        fm_link:
00002FF3 B80258             mov     ax,DOS_CHECK_UMBLINK ; 5802h
00002FF6 CD21               int     21h                 ; Current link-state is now in al

                                ;putdata fm_umb,al          ; So store it in fm_umb for later
                                ;
                                ;push   es
                                ;push   cs
                                ;pop    es
```

```
                                    ;mov    [es:fm_umb],al
                                    ;pop    es

                                    ; 01/01/2023
                                    ; ds = cs
                                    ;mov    [cs:fm_umb],al
00002FF8 A2[3024]                   mov     [fm_umb],al

00002FFB B80358                     mov     ax,DOS_SET_UMBLINK ; 5803h
00002FFE BB0100                     mov     bx,1
00003001 CD21                       int     21h
00003003 C3                         retn

                                ; ----------------------------------------------------------------------------
                                ;*** fm_unlink - unlinks UMBs if fm_umb is set to 0
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:    fm_umb == 1 : leave linked, else unlink
                                ; EXIT:     None
                                ; ERROR:    None
                                ; USES:     AX, BX
                                ; ----------------------------------------------------------------------------

                                    ; 01/01/2023 - Retro DOS v4.2
                                fm_unlink:
00003004 31DB                       xor     bx,bx

                                    ;getdata bl,fm_umb           ; fm_umb already has the old link-state
                                    ;
                                    ;push   ds
                                    ;push   cs
                                    ;pop    ds
                                    ;mov    bl,[fm_umb]
                                    ;pop    ds

                                    ; 01/01/2023
                                    ; ds = cs
                                    ;mov    bl,[cs:fm_umb]
00003006 8A1E[3024]                 mov     bl,[fm_umb]

0000300A B80358                     mov     ax,DOS_SET_UMBLINK ; 5803h
0000300D CD21                       int     21h                 ; so just use that, and call int 21h
0000300F C3                         retn

                                ; 08/04/2019 - Retro DOS v4.0

                                ; ----------------------------------------------------------------------------
                                ;*** ParseVar - parses [/S][/L:umb[,size][;umb[,size]]*] and builds the table
                                ; laid out in highvar.inc
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:    ES:SI points to command tail of LoadHigh/DeviceHigh (whitespace ok)
                                ; EXIT:     ES:SI points to first character in child program name
                                ; ERROR:    ES:SI points to character which caused error, carry set, AX == code
                                ; USES:     ES:SI, AX, flags, variables in highvar.inc
                                ; ----------------------------------------------------------------------------
                                ; Error codes (in AX if carry set on return):
                                ;
                                PV_InvArg   equ     1       ; Invalid argument passed
                                PV_BadUMB   equ     2       ; Bad UMB number passed (duplicate?)
                                PV_InvSwt   equ     3       ; Unrecognized switch passed
                                ;
                                ; This routine exects ES:SI to point to a string much like the following:
                                ;    "/S/L:1,200;2 module options"
                                ; Optionally, the string can begin with whitespace; neither /S nor /L is
                                ; required, though that's what this routine is supposed to parse.
                                ;
                                optS        equ     'S'     ; /S
                                optL        equ     'L'     ; /L:...
                                ;
                                ; ----------------------------------------------------------------------------
                                ; LoadHigh has a list of arguments, returned by cparse, which is used to create
                                ; a command-line for spawning a child process. For a typical LH command, say,
                                ;     lh /l:1,1000;2 print/d:lpt2
                                ; the arguments would look like (one per line):
                                ;     lh
                                ;     /l
                                ;     1
                                ;     1000
                                ;     2
                                ;     print
                                ;     /d
                                ;     :lpt2
                                ; In short, if "print" were, say, "43", there'd be no way to determine which
                                ; arg was the filename. So, inside this routine, we keep a running counter
                                ; of the number of arguments LH will need to skip in order to get to the
                                ; program name. The "lh" is implicit--it'll always have to skip that. So if
                                ; there's no "/l" or "/s", fm_argc will be 0 ... other than that, 1 is added
                                ; for:
                                ;    Each /L
                                ;    Each /S (there should be only one)
                                ;    Each UMB number (they follow ":" or ";")
                                ;    Each UMB size   (they follow ",")
                                ; So, in the above example, fm_argc would be 4-- and LH would skip right to
                                ; "print".  Note that InitVar initializes fm_argc to zero.
                                ; ----------------------------------------------------------------------------

                                    ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:2F9Fh)

                                ParseVar:
                                    ;pushreg <di, ds, es>
                                    ; 01/01/2023
                                    ;push   di ; * ; (not required) ; 01/01/2023
00003010 1E                         push    ds
00003011 06                         push    es

00003012 06                         push    es              ; Make DS:SI point to it, as well as ES:SI
00003013 1F                         pop     ds              ; (regardless if we're in devhigh or loadhigh)
00003014 FC                         cld

                                ; ------------------------------------------------
                                ; PV10--ES:SI = any whitespace on the command-line
                                ; ------------------------------------------------

00003015 AC                     pv10:       lodsb                   ; here, ES:SI==" /L..."--must eat whitespace
00003016 E8A200                     call    isWhite
00003019 74FA                       jz      short pv10      ;       ES:SI==" /L..."--keep eating.
                                    ;cmp    al,'/'
0000301B 3C2F                       cmp     al,SWTCH
0000301D 7404                       je      short pv20      ;       ES:SI=="/L..."--go process a switch

0000301F 4E                         dec     si              ; Backup--it's now "odule options", and we need
00003020 F8                         clc                     ; that "m" we just read (or whatever it is).
00003021 EB2B                       jmp     short pvX       ; Then return with carry clear == we're done.

```

```
00003023 AC                     pv20:       lodsb                   ; Just read 'S' or 'L', hopefully
                                    ;toUpper al             ; So we make it upper-case, and...
00003024 24DF                       and     al,0DFh
                                    ;cmp    al,'S'
00003026 3C53                       cmp     al,optS         ; just read 'S'?
00003028 750D                       jne     short pv30
                                    ;call   incArgc         ; If it's /S, it's another arg for LH to skip.
0000302A 2EFE06[3224]               inc     byte [cs:fm_argc] ; 19/04/2019
                                    ;putdata fUmbTiny,1    ; /S, so ES:SI=="  /L..." or " module opts", or
                                    ;
                                    ;push   es
                                    ;push   cs
                                    ;pop    es
                                    ;mov    [es:fUmbTiny],1
                                    ;pop    es
0000302F 2EC606[FC23]01             mov     byte [cs:fUmbTiny],1
00003035 EBDE                       jmp     short pv10      ; possibly even "/L...".
                                pv30:       ;cmp    al,'L'
00003037 3C4C                       cmp     al,optL         ; If it's not 'L' either, then 'tis a bad
00003039 750D                       jne     short pvE1              ; switch!
                                    ;call   incArgc         ; If it's /L, it's another arg for LH to skip.
0000303B 2EFE06[3224]               inc     byte [cs:fm_argc] ; 19/04/2019
00003040 E80E00                     call    parseL
00003043 73D0                       jnc     short pv10      ; If no carry, go back and look for more
00003045 4E                         dec     si              ; Else, back up and exit.
00003046 EB03                       jmp     short pvErr     ; AX has already been set by parseL
                                pvE1:       ;mov    ax,3
00003048 B80300                     mov     ax,PV_InvSwt    ; Unrecognized switch passed
0000304B 4E                     pvErr:      dec     si
0000304C 4E                         dec     si
0000304D F9                         stc
                                pvX:;popreg <es, ds, di>
0000304E 07                         pop     es
0000304F 1F                         pop     ds
                                    ; 01/01/2023
                                    ;pop    di ; * ; (not required) ; 01/01/2023
00003050 C3                         retn

                                ; ---------------------------------------------------------------------------
                                ;*** parseL - parses ":nnnn[,nnnn][;nnnn[,nnnn]]*" for ParseVar
                                ; ---------------------------------------------------------------------------
                                ; ENTRY:    ES:SI points to colon
                                ; EXIT:     ES:SI points to first character not parsed
                                ; ERROR:    Carry set; rewind three characters and return (see ParseVar)
                                ; USES:     ES:SI, flags, AX, CX, DX, variables in highvar.inc
                                ; ---------------------------------------------------------------------------
                                ; If the string here is terminated with anything other than whitespace or a
                                ; switchchar (perhaps it's /S or another /L:... ), then we return with carry
                                ; set, indicating that they've screwed up the syntax.  The 3-character rewind
                                ; makes sure the app /L: is reported as being the culprit.
                                ; ---------------------------------------------------------------------------

                                parseL:
00003051 AC                         lodsb
00003052 3C3A                       cmp     al,':'          ; Make sure they did /L:
00003054 754E                       jne     short plE1      ; If they didn't, return with carry set.

                                ; ----------------------------------------
                                ; PL10--ES:SI = a UMB number, after /L: or ;
                                ; ----------------------------------------

00003056 E8DB00                 pl10:       call    GetXNum         ; After this, 'tis ",size" or ";umb" or " mod"
00003059 724F                       jc      short plE2      ; And error if it's a bad number.
0000305B E89D01                     call    convUMB         ; Convert any address to a UMB number

0000305E 88C1                       mov     cl,al           ; Remember the UMB number
00003060 E87600                     call    stowUMB         ; Mark this UMB # as used;
00003063 7245                       jc      short plE2      ; If it was already marked, it'll error

                                    ;call   incArgc         ; Each UMB number is another arg for LH to skip
00003065 2EFE06[3224]               inc     byte [cs:fm_argc] ; 08/04/2019 - Retro DOS v4.0

0000306A AC                         lodsb
0000306B 3C3B                       cmp     al,';'          ; Did "umb;" ?
0000306D 74E7                       je      short pl10      ; Yep: go back and get another UMB.

0000306F E84900                     call    isWhite         ; Did "umb " ?
00003072 743B                       jz      short plX       ; Yep: return (it'll go back to whitespace)

00003074 E83900                     call    isEOL           ; Did "umb" ?
00003077 7435                       jz      short plSwX     ; If so, backup and exit like everything's ok

                                    ;cmp    al,'/'
00003079 3C2F                       cmp     al,SWTCH        ; Did "umb/" ? (as in, "/L:1,100;2/S")
0000307B 7431                       je      short plSwX     ; If so, back up ES:SI one character and return

0000307D 3C2C                       cmp     al,','          ; Did "umb," ?
0000307F 7523                       jne     short plE1      ; Just what the heck DID they do? Return error.

                                ; --- Read a size -----------------------------------------------------------

00003081 E8B000                     call    GetXNum         ; Stop on "size;" or "size " or anything else
00003084 721E                       jc      short plE1      ; And error if it's a bad size.

00003086 E81601                     call    toPara          ; Convert from bytes to paragraphs

00003089 E8CE01                     call    stowSiz         ; CL still has the UMB number for this routine

                                    ;call   incArgc         ; Each UMB size is another arg for LH to skip
0000308C 2EFE06[3224]               inc     byte [cs:fm_argc] ; 08/04/2019 - Retro DOS v4.0

00003091 AC                         lodsb
00003092 3C3B                       cmp     al,';'          ; They did "umb,size;", so get another UMB.
00003094 74C0                       je      short pl10      ;

00003096 E82200                     call    isWhite         ; Did it end with whitespace?
00003099 7414                       jz      short plX       ; If so, we're done here--go back.

0000309B E81200                     call    isEOL           ; Did they do "umb,size" and end??? (stupid)
0000309E 740E                       jz      short plSwX     ; If so, backup and exit like everything's ok

                                    ;cmp    al,'/'
000030A0 3C2F                       cmp     al,SWTCH        ; Did they do "umb,size/" ?
000030A2 740A                       je      short plSwX     ; If so, again, we're done here.
                                plE1:
```

```
                                    ;mov    ax,1
000030A4 B80100                     mov     ax,PV_InvArg   ; If not, we don't know WHAT they did...
000030A7 4E                         dec     si
000030A8 F9                         stc
000030A9 C3                         retn

                                plE2:       ;mov    ax,2
000030AA B80200                     mov     ax,PV_BadUMB   ; In this case, they've specified a UMB twice
                                    ; 12/12/2022
                                    ; cf=1
                                    ;stc
000030AD C3                         retn
                                plSwX:
000030AE 4E                         dec     si             ; If we hit a '/' character, back up one char
                                                           ; so the whitespace checker will see it too.
                                plX:; 12/12/2022
                                    ; cf=0
                                    ;clc                   ; Then just return with carry clear, so
000030AF C3                         retn                   ; ParseVar will go about its business.

                                ; ---------------------------------------------------------------------------
                                ;*** incArgc - increments fm_argc, for use with LoadHigh command-line parsing
                                ; ---------------------------------------------------------------------------
                                ; ENTRY:    None
                                ; EXIT:     None
                                ; ERROR:    None
                                ; USES:     fm_argc, flags
                                ; ---------------------------------------------------------------------------

                                ;incArgc:
                                    ;push  ax

                                    ;;getdata al, fm_argc ; Obtain previous value of fm_argc,

                                    ;mov   al,[cs:fm_argc]

                                    ;inc   al              ; Increment it,

                                    ;;putdata fm_argc, al ; And store it right back.

                                    ;mov   [cs:fm_argc],al

                                    ;pop   ax
                                    ;retn

                                ; ---------------------------------------------------------------------------
                                ;*** isEOL - returns with ZF set if AL contains CR or LF, or 0
                                ; ---------------------------------------------------------------------------
                                ; ENTRY:    AL contains character to test
                                ; EXIT:     ZF set iff AL contains CR or LF, or 0
                                ; ERROR:    None
                                ; USES:     ZF
                                ; ---------------------------------------------------------------------------

                                isEOL:
000030B0 3C00                       cmp     al,0           ; Null-terminator
000030B2 7406                       je      short ieX
000030B4 3C0D                       cmp     al,CR ; 0Dh    ; Carriage Return
000030B6 7402                       je      short ieX
000030B8 3C0A                       cmp     al,LF ; 0Ah    ; LineFeed
                                ieX:
000030BA C3                         retn

                                ; ---------------------------------------------------------------------------
                                ;*** isWhite - returns with ZF set if AL contains whitespace (or "=")
                                ; ---------------------------------------------------------------------------
                                ; ENTRY:    AL contains character to test
                                ; EXIT:     ZF set iff AL contains space, tab, or equals
                                ; ERROR:    None
                                ; USES:     ZF
                                ; ---------------------------------------------------------------------------

                                isWhite:
000030BB 3C20                       cmp     al,' '         ; Space
000030BD 7406                       je      short iwX
000030BF 3C3D                       cmp     al,'='         ; Equals (treat as whitespace)
000030C1 7402                       je      short iwX
000030C3 3C09                       cmp     al,tab ; 9     ; Tab
                                iwX:
000030C5 C3                         retn

                                ; ---------------------------------------------------------------------------
                                ;*** unMarkUMB - marks a given UMB as unused, even if previously marked used
                                ; ---------------------------------------------------------------------------
                                ; ENTRY:    AL contains UMB number
                                ; EXIT:     None
                                ; ERROR:    None
                                ; USES:     Flags, variables in highvar.inc
                                ; ---------------------------------------------------------------------------

                                    ; 02/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)

                                unMarkUMB:
                                    ; 02/01/2023
                                    ;push  ax
                                    ;push  bx
                                    ;push  di
                                    ;push  es
                                    ;
                                    ;push  cs
                                    ;pop   es

000030C6 30E4                       xor     ah,ah
000030C8 89C3                       mov     bx,ax

                                    ; 19/04/2019

                                    ;;mov  byte [es:bx+UmbUsed],0
                                    ;mov   [es:bx+UmbUsed],ah ; 0
                                    ; 02/01/2023
                                    ; ds= cs
                                    ;mov   [cs:bx+UmbUsed],ah ; 0
000030CA 88A7[0024]                 mov     [bx+UmbUsed],ah ; 0

000030CE 3806[FF23]                 cmp     [UmbLoad],al
                                    ;cmp   [cs:UmbLoad],al
                                    ;;cmp  [es:UmbLoad],al
000030D2 7504                       jne     short umu10

                                    ;;mov  [es:UmbLoad],0 ; If unmarked the load UMB, load into convent.
                                    ;mov   [es:UmbLoad],ah ; 0
                                    ; 02/01/2023
                                    ; ds = cs
                                    ;mov   [cs:UmbLoad],ah ; 0
```

```
37785 000030D4 8826[FF23]               mov     [UmbLoad],ah ; 0
37786                                  umu10:
37787                                      ;pop    es
37788                                      ;pop    di
37789                                      ;pop    bx
37790                                      ;pop    ax
37791 000030D8 C3                          retn
37792
37793                                  ; ---------------------------------------------------------------------------
37794                                  ;*** stowUMB - marks a given UMB as used, if it hasn't been so marked before
37795                                  ;              -- accepts a UMB # in AL, and makes sure it hasn't yet been
37796                                  ; listed in the /L:... chain. If it's the first one specified, it sets UmbLoad
37797                                  ; to that UMB #... and in any case, it marks the UMB as specified.
37798                                  ; ---------------------------------------------------------------------------
37799                                  ; ENTRY:    AL contains UMB number, as specified by the user
37800                                  ; EXIT:     None
37801                                  ; ERROR:    Carry set if UMB # is less than 0 or >= MAXUMB (see highvar.inc)
37802                                  ; USES:     AX, Flags, variables in highvar.inc
37803                                  ; ---------------------------------------------------------------------------
37804
37805                                      ; 01/01/2023 - Retro DOS v4.2
37806                                  stowUMB:
37807 000030D9 3C10                      cmp     al,MAXUMB ; 16
37808 000030DB 7202                      jb      short su10
37809 000030DD F9                        stc
37810 000030DE C3                        retn                    ; Ooops-- UMB>=MAXUMB
37811                                  su10:
37812                                      ; 01/01/2023
37813                                      ;push   bx
37814                                      ;push   di
37815                                      ;push   si
37816                                      ;push   ds
37817                                      ;push   es
37818                                      ;push   cs
37819                                      ;pop    es
37820                                      ;push   cs
37821                                      ;pop    ds
37822
37823                                      ; 01/01/2023
37824                                      ; ds <> cs
37825                                      ;cmp    byte [cs:UmbLoad],0FFh
37826 000030DF 2E803E[FF23]FF            cmp     byte [cs:UmbLoad],UNSPECIFIED
37827                                                              ; If this, we haven't been here before
37828 000030E5 7504                      jne     short su20
37829 000030E7 2EA2[FF23]                mov     [cs:UmbLoad],al       ; So remember this UMB as the load UMB slot.
37830
37831                                      ;;cmp   byte [UmbLoad],0FFh
37832                                      ;cmp    byte [UmbLoad],UNSPECIFIED ; If this, we haven't been here before
37833                                      ;jne    short su20
37834                                      ;mov    [UmbLoad],al   ; So remember this UMB as the load UMB slot.
37835                                  su20:
37836 000030EB 08C0                      or      al,al           ; If they gave UMB 0, there's really nothing
37837 000030ED 740E                      jz      short su30      ; that we should do here.
37838
37839                                      ;mov    bl,al
37840                                      ;xor    bh,bh
37841                                      ;mov    ax,1            ; Now, AX = 1, and BX = UMB Number
37842                                      ; 01/01/2023
37843 000030EF 30E4                      xor     ah,ah
37844 000030F1 89C3                      mov     bx,ax
37845 000030F3 B001                      mov     al,1
37846
37847                                      ;xchg   [es:bx+UmbUsed],al
37848                                      ; 01/01/2023
37849 000030F5 2E8687[0024]              xchg    [cs:bx+UmbUsed],al
37850
37851                                      ;or     al,al           ; If it was already 1, then al==1... and that
37852                                      ;jz     short su30      ; means an error.
37853                                      ;
37854                                      ;stc                    ; OOOPS!  This one's been used before.  :(
37855
37856                                      ; 01/01/2023
37857 000030FA 3C01                      cmp     al,1
37858 000030FC F5                        cmc     ; if al > 0 -> cf = 1
37859                                  su30:
37860                                      ; 01/01/2023
37861                                      ;pop    es
37862                                      ;pop    ds
37863                                      ;pop    si
37864                                      ;pop    di
37865                                      ;pop    bx
37866 000030FD C3                        retn
37867
37868                                  ; 08/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
37869                                  %if 0
37870                                  ; ---------------------------------------------------------------------------
37871                                  ;*** stowSiz - marks a given UMB as having a given minimum size
37872                                  ; ---------------------------------------------------------------------------
37873                                  ; ENTRY:    CL contains UMB number, AX contains size
37874                                  ; EXIT:     None
37875                                  ; ERROR:    None
37876                                  ; USES:     AX, DX, Flags, variables in highvar.inc
37877                                  ; ---------------------------------------------------------------------------
37878
37879                                  ; 13/05/2019
37880
37881                                      ; 01/01/2023 - Retro DOS v4.2
37882                                  stowSiz:
37883                                      ; 01/01/2023
37884                                      ;push   bx
37885                                      ;;push  di ; ?
37886                                      ;push   es
37887
37888                                      ;push   cs
37889                                      ;pop    es
37890
37891                                      mov     bl,cl                   ; Now bl==UMB number, AX==size
37892                                      mov     bh,0                    ;     bx==UMB number, AX==size
37893                                      shl     bl,1                    ;     bx==offset into array, AX=size
37894                                      ;mov    [es:bx+UmbSize],ax      ; Store the size
37895                                      ; 01/01/2023
37896                                      mov     [cs:bx+UmbSize],ax      ; Store the size
37897
37898                                      ; 01/01/2023
37899                                      ;pop    es
37900                                      ;;pop   di ; ?
37901                                      ;pop    bx
37902
37903                                      retn
37904                                  %endif
37905
37906                                  ; ---------------------------------------------------------------------------
37907                                  ;*** toDigit - converts a character-digit to its binary counterpart
37908                                  ;              -- verifies that CL contains a valid character-digit; if so, it
```

```
                                ; changes CL to its counterpart binary digit ((CL-'0') or (CL-'A'+10)).
                                ; A-F are considered valid iff gnradix is 16.
                                ; ---------------------------------------------------------------------------
                                ; ENTRY:    CL contains a digit ('0' to '9' or, if gnradix==16, 'A' to 'F')
                                ; EXIT:     CL contains digit in binary (0 to 9 or, if gnradix==16, 0 to 15)
                                ; ERROR:    Carry set indicates invalid digit; carry clear indicates good digit
                                ; USES:     CL, Flags
                                ; ---------------------------------------------------------------------------
                                ; If the string is preceeded with "0x", the value is read as hexadecimal; else,
                                ; as decimal. After a read, you may check the radix by examining gnradix--it
                                ; will be 10 or 16.
                                ; ---------------------------------------------------------------------------

                                gnradix:
000030FE 0000                       dw      0               ; Must be a word--16x16 multiplication

                                toDigit:
00003100 2E833E[FE30]10             cmp     word [cs:gnradix],16
00003106 751C                       jne     short td20     ; Don't check hex digits if radix isn't 16

                                toDigit_hex:
00003108 80F961                     cmp     cl,'a'  ; 61h
0000310B 7209                       jb      short td10
0000310D 80F966                     cmp     cl,'f'  ; 66h
00003110 7720                       ja      short tdE      ; Nothing valid above 'z' at all...
00003112 80E957                     sub     cl,'a'-10 ; 57h       ; Make 'a'==10 and return.
                                    ;clc                    ; <- CLC is implicit from last SUB
00003115 C3                         retn
                                td10:
00003116 80F941                     cmp     cl,'A'  ; 41h
00003119 7209                       jb      short td20     ; Below 'A'?  Not a letter...
0000311B 80F946                     cmp     cl,'F' ; 46h
0000311E 7712                       ja      short tdE      ; Above 'F'?  Not a digit.
00003120 80E937                     sub     cl,'A'-10 ; 37h       ; Make 'A'==10 and return.
                                    ;clc                    ; <- CLC is implicit from last SUB
00003123 C3                         retn
                                toDigit_dec:
                                td20:
00003124 80F930                     cmp     cl,'0'         ; If less than zero,
                                    ;jb     short tdE      ; Done.
00003127 720A                       jb      short tdEr ; 08/04/2019
00003129 80F939                     cmp     cl,'9'         ; Or, if greater than nine,
0000312C 7704                       ja      short tdE      ; Done.
0000312E 80E930                     sub     cl,'0'  ; 30h   ; Okay--make '0'==0 and return.
                                    ;clc                    ; <- CLC is implicit from last SUB
00003131 C3                         retn
                                tdE:
00003132 F9                         stc
                                tdEr:               ; 08/04/2019 - Retro DOS v4.0
00003133 C3                         retn

                                ; ---------------------------------------------------------------------------
                                ;*** GetXNum - reads a 32-bit ASCII number at ES:SI and returns it in DX:AX
                                ; ---------------------------------------------------------------------------
                                ; ENTRY:    ES:SI points to an ascii string to scan
                                ; EXIT:     ES:SI moved to first invalid digit, DX:AX contains value read
                                ; ERROR:    Carry set if # is too big, or has no digits (EOL possibly)
                                ; USES:     ES:SI, DX, AX, Flags, gnradix
                                ; ---------------------------------------------------------------------------
                                ; If the string is preceeded with "0x", the value is read as hexadecimal; else,
                                ; as decimal. After a read, you may check the radix by examining gnradix--it
                                ; will be 10 or 16.
                                ; ---------------------------------------------------------------------------

                                ; 08/04/2019 - Retro DOS v4.0

                                ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:3109h)

                                GetXNum:
                                    ;pushreg <bx, cx, ds>
                                    ; 01/01/2023
                                    ;push   bx
00003134 51                         push    cx ; *
                                    ;push   ds

00003135 FC                         cld
00003136 31C0                       xor     ax,ax
00003138 31DB                       xor     bx,bx
0000313A 31C9                       xor     cx,cx
0000313C 31D2                       xor     dx,dx                   ; Start with 0 (makes sense)

0000313E 2EC706[FE30]0A00           mov     word [cs:gnradix],10  ; And default to a radix of 10 (dec)

00003145 268A0C                     mov     cl,[es:si]              ; Now AX=0, BX=0, CH=0/CL=char, DX=0
                                    ;call   toDigit
00003148 E8D9FF                     call    toDigit_dec
                                    ;jc     short gxnE              ; If it's not a digit, leave now.
                                    ; 01/01/2023
0000314B 7233                       jc      short gxnX

0000314D 08C9                       or      cl,cl
0000314F 7517                       jnz     short gxn20             ; Doesn't have '0x'
00003151 268A4C01                   mov     cl,[es:si+1]
00003155 80F978                     cmp     cl,'x'                  ; Either 'x'...
00003158 7405                       je      short gxn10
0000315A 80F958                     cmp     cl,'X'                  ; ...or 'X' means it's hexadecimal
0000315D 7509                       jne     short gxn20

                                gxn10:
0000315F 2EC706[FE30]1000           mov     word [cs:gnradix], 16
00003166 46                         inc     si                      ; Since we read "0x", march over it.
00003167 46                         inc     si

                                ; ---------------------------------------------------
                                ; GXN20--ES:SI = a digit in a number; if not, we're done
                                ;       DX:AX = current total
                                ;       BX    = 0
                                ;       CH    = 0
                                ; ---------------------------------------------------

                                gxn20:
00003168 268A0C                     mov     cl,[es:si]      ; Now DX:AX=current total, CH=0/CL=char
0000316B 46                         inc     si

0000316C E891FF                     call    toDigit         ; Accepts only valid digits, A-F -> 10-16
0000316F 720D                       jc      short gxnQ      ; <- Ah... wasn't a digit. Stop.

00003171 E80E00                     call    mul32           ; Multiply DX:AX by gnradix
00003174 720A                       jc      short gxnX      ; (if it's too big, error out)

00003176 01C8                       add     ax,cx           ; Add the digit
00003178 11DA                       adc     dx,bx           ; (BX is 0!)--Adds 1 iff last add wrapped
                                    ;jc     short gxnX      ; If _that_ wrapped, it's too big.
```

```
38033                                   ;jmp    short gxn20
38034 0000317A 73EC                     jnc     short gxn20
38035                                   gxnE:
38036                                   ;stc                    ; In this case, we need to set the carry
38037 0000317C EB02                     jmp     short gxnX      ; and leave--there were no digits given.
38038                                   gxnQ:
38039 0000317E 4E                       dec     si              ; Don't read in the offensive character.
38040 0000317F F8                       clc                     ; And clear carry, so they know it's okay.
38041                                   gxnX:
38042                                   ; 01/01/2023
38043                                   ;pop    ds
38044 00003180 59                       pop     cx ; *
38045                                   ;pop    bx
38046 00003181 C3                       retn
38047
38048                                   ; ---------------------------------------------------------------------------
38049                                   ;*** mul32 - multiplies the number in DX:AX by gnradix
38050                                   ; ---------------------------------------------------------------------------
38051                                   ; ENTRY:   DX:AX = the number to be multiplied, BX = 0, gnradix = multiplier
38052                                   ; EXIT:    DX:AX has been multiplied by gnradix if carry clear; BX still 0
38053                                   ; ERROR:   Carry set if number was too large
38054                                   ; USES:    Flags, AX, DX
38055                                   ; ---------------------------------------------------------------------------
38056
38057                                   mul32:
38058 00003182 50                       push    ax              ; DX=old:hi, AX=old:lo, TOS=old:lo, BX=0
38059 00003183 89D0                     mov     ax,dx           ; DX=old:hi, AX=old:hi, TOS=old:lo, BX=0
38060 00003185 2EF726[FE30]             mul     word [cs:gnradix] ; DX=?, AX=new:hi, TOS=old:lo, BX=0
38061 0000318A 7211                     jc      short m32E      ; Too big?
38062
38063 0000318C 89C2                     mov     dx,ax           ; DX=new:hi, AX=new:hi, TOS=old:lo, BX=0
38064 0000318E 58                       pop     ax              ; DX=new:hi, AX=old:lo, TOS=orig, BX=0
38065
38066 0000318F 87DA                     xchg    dx,bx           ; DX=0, AX=old:lo, TOS=orig, BX=new:hi
38067 00003191 2EF726[FE30]             mul     word [cs:gnradix] ; DX=carry,  AX=new:lo, TOS=orig, BX=new:hi
38068 00003196 87DA                     xchg    dx,bx           ; DX=new:hi, AX=new:lo, TOS=orig, BX=carry
38069 00003198 01DA                     add     dx,bx           ; DX=new:hi, AX=new:lo, TOS=orig, BX=carry
38070 0000319A 31DB                     xor     bx,bx           ; DX=new:hi, AX=new:lo, TOS=orig, BX=0
38071 0000319C C3                       retn
38072                                   m32E:
38073 0000319D 58                       pop     ax
38074 0000319E C3                       retn
38075
38076                                   ; ---------------------------------------------------------------------------
38077                                   ;*** toPara - divides DX:AX by 16; result in AX only (discards extra DX data)
38078                                   ; ---------------------------------------------------------------------------
38079                                   ; ENTRY:   DX:AX = the number to be divided
38080                                   ; EXIT:    Interpereting DX:AX as bytes, AX=paragraph equivalent, 0xFFFF max
38081                                   ; ERROR:   None
38082                                   ; USES:    Flags, AX, DX
38083                                   ; ---------------------------------------------------------------------------
38084                                   ; Note: The 386 has a 32-bit SHR, which would work perfectly for this... but we
38085                                   ;       can't ensure a 386 host machine. Sorry.
38086                                   ; ---------------------------------------------------------------------------
38087
38088                                       ; 01/01/2023 - Retro DOS v4.2
38089                                   toPara:
38090 0000319F 51                       push    cx              ; DX:AX=HHHH hhhh hhhh hhhh:LLLL llll llll llll
38091
38092 000031A0 B104                     mov     cl,4            ;
38093 000031A2 D3E8                     shr     ax,cl           ; DX:AX=HHHH hhhh hhhh hhhh:0000 LLLL llll llll
38094 000031A4 92                       xchg    ax,dx           ; DX:AX=0000 LLLL llll llll:HHHH hhhh hhhh hhhh
38095 000031A5 B10C                     mov     cl,12
38096 000031A7 D3E0                     shl     ax,cl           ; DX:AX=0000 LLLL llll llll:hhhh 0000 0000 0000
38097 000031A9 09D0                     or      ax,dx           ;    AX=hhhh LLLL llll llll
38098
38099 000031AB 59                       pop     cx
38100 000031AC C3                       retn
38101
38102                                   ; ---------------------------------------------------------------------------
38103                                   ;*** UmbHead - returns in AX the address of the first UMB block (0x9FFF)
38104                                   ; ---------------------------------------------------------------------------
38105                                   ; ENTRY:  Nothing
38106                                   ; EXIT:   AX contains 0x9FFF for most systems
38107                                   ; ERROR:  Carry set if pointer is 0xFFFF (if not set up yet--DH runs into this)
38108                                   ; USES:   Flags, AX
38109                                   ; ---------------------------------------------------------------------------
38110                                   ; Early in the boot-cycle, the pointer used to obtain this value isn't set up;
38111                                   ; to be precise, before a UMB provider is around. In this event, the pointer
38112                                   ; is always set to 0xFFFF; it changes once a provider is around. On most
38113                                   ; machines (all of 'em I've seen), it changes to 0x9FFF at that point.
38114                                   ; ---------------------------------------------------------------------------
38115
38116                                       ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
38117                                   UmbHead:
38118                                       ; 13/05/2019 (because of callers, pushs & pops are not needed here)
38119
38120                                       ;push   si ; ?
38121                                       ;push   ds ; ?
38122                                       ;push   es
38123                                       ;push   bx ; *
38124
38125                                       ; 09/04/2019
38126                                       ; !!! No need to save es,bx,ds,si above !!! (es,bx are changed here)
38127
38128 000031AD B452                       mov     ah,GET_IN_VARS          ; Call int 21h, function 52h...
38129 000031AF CD21                       int     21h
38130
38131 000031B1 26A18C00                   mov     ax,[es:DOS_UMB_HEAD]    ; And read what's in ES:[008C]
38132
38133                                       ; 01/01/2023
38134 000031B5 83F8FF                     cmp     ax,0FFFFh
38135 000031B8 F5                         cmc
38136                                       ; if AX=0FFFFh -> CF=1
38137 000031B9 C3                         retn
38138
38139                                   ; 01/01/2023
38140                                   ;%if 0
38141                                   ;   cmp     ax,0FFFFh
38142                                   ;   je      short uhE               ; If it's 0xFFFF, it's an error...
38143                                   ;
38144                                   ;   clc                             ; Else, it isn't (CLC done by prev cmp)
38145                                   ;   ;jmp    short uhX
38146                                   ;   ; 12/12/2022
38147                                   ;   retn
38148                                   ;uhE:
38149                                   ;   stc
38150                                   ;uhX:
38151                                   ;   ;pop    bx ; *
38152                                   ;   ;pop    es
38153                                   ;   ;pop    ds ; ?
38154                                   ;   ;pop    si ; ?
38155                                   ;   retn
38156                                   ;%endif
```

```
                                ; ----------------------------------------------------------------------------
                                ;*** isSysMCB - sets ZF if ES points to an MCB owned by "SC" + (8 or 9)
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:  ES:0 should point to a valid MCB
                                ; EXIT:   ZF set if owned by SC+8 or SC+9 (for japan)
                                ; USES:   Flags
                                ; ----------------------------------------------------------------------------

                                isSysMCB:
                                    ;push  ax

                                    ;mov   ax,[es:ARENA.OWNER]   ; Check the owner...
                                    ;cmp   ax,SystemPSPOwner     ; 8 (for US OR Japan) is valid
                                    ;je    short ism10
                                    ;cmp   ax,JapanPSPOwner      ; 9 (for Japan) is valid
                                    ;;je   short ism10
                                    ;;jmp  short ismX            ; Anything else isn't.
                                    ;jne   short ismX
000031BA 26833E010008               cmp    word [es:ARENA.OWNER],SystemPSPOwner ; 8  ; 09/04/2019
000031C0 7507                       jne    short ismX
                                ism10:
                                    ;mov   ax,[es:ARENA.NAME]    ; Check the name...
                                    ;cmp   ax,'SC' ; 4353h
000031C2 26813E08005343             cmp    word [es:ARENA.NAME],'SC'
                                ismX:
                                    ;pop   ax
000031C9 C3                         retn

                                ; 09/04/2019 - Retro DOS v4.0

                                ; ----------------------------------------------------------------------------
                                ;*** AddrToUmb - converts a segment address in AX to its appropriate UMB number
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:  AX contains a segment address
                                ; EXIT:   AX will contain the UMB number which contains the address (0==conv)
                                ; ERROR:  If the address is above UM Range, AX will return as FFFF.
                                ; USES:   Flags, AX
                                ; ----------------------------------------------------------------------------
                                ; An address in the following areas is treated as:
                                ;    0      <-> umbhead (0x9FFF)          = Conventional memory
                                ;    0x9FFF <-> addr of first UM sys MCB  = UMB #1
                                ;      ...
                                ;    addr of last UM sys MCB <-> TOM      = invalid; returns #0xFFFF
                                ; ----------------------------------------------------------------------------

                                    ; 01/01/2023 - Retro DOS v4.2
                                AddrToUmb:
                                    ; 01/01/2023
                                    ;push  cx
                                    ;push  dx
000031CA 06                         push   es

000031CB 89C2                       mov    dx,ax          ; DX = address to search for

000031CD E8DDFF                     call   UmbHead        ; AX = first segment
000031D0 7222                       jc     short atuE     ; If it couldn't get it, error out.

                                    ; 22/07/2023
                                    ;mov   es,ax ; *      ; ES = first UMB segment
000031D2 31C9                       xor    cx,cx ; 0      ; Pretend we're on UMB 0 for now... (cx = UMB#)

                                    ; 22/07/2023
                                atu10:
000031D4 8EC0                       mov    es,ax ; * ; ** ; 22/07/2023
                                ; ----------------------------------------
                                ; ATU10--ES - Current MCB address
                                ;        DX - Address given for conversion
                                ;        CX - Current UMB #
                                ; ----------------------------------------

                                ;atu10:
                                    ;mov   ax,es ; * ; 18/07/2023
000031D6 39D0                       cmp    ax,dx          ; Present segment >= given segment?
000031D8 731D                       jae    short atuX     ; Yep--done.

000031DA E8DDFF                     call   isSysMCB       ; Returns with ZF set if this is a system MCB
000031DD 7501                       jnz    short atu20

000031DF 41                         inc    cx             ; If it _was_ a system MCB, we're in a new UMB.
                                atu20:
                                    ;mov   al,[es:ARENA.SIGNATURE]
                                    ;cmp   al,arena_signature_end  ; 'Z'
                                    ; 22/07/2023
                                    ; ax = es
                                    ;mov   ax,es ; **
000031E0 2603060300                 add    ax,[es:ARENA.SIZE]
000031E5 26803E00005A               cmp    byte [es:ARENA.SIGNATURE],arena_signature_end
000031EB 7403                       je     short atu30           ; 'Z' means this was the last MCB... that's it.

                                    ;NextMCB es,ax

                                    ;mov   ax,es ; **
                                    ;;add  ax,[es:3]
                                    ;add   ax,[es:ARENA.SIZE]
000031ED 40                         inc    ax
                                    ; 22/07/2023
                                    ;mov   es,ax ; *
000031EE EBE4                       jmp    short atu10

                                ; ----------------------------------------------------------------------------
                                ; if we get to atu30, they specified a number that was past the last MCB.
                                ; make sure it's not _inside_ that MCB before we return an error condition.
                                ; ----------------------------------------------------------------------------

                                atu30:
                                    ; 22/07/2023
                                    ; ax = es + [es:ARENA.SIZE]
                                    ;mov   ax,es ; **
                                    ;add   ax,[es:ARENA.SIZE] ; **
000031F0 39D0                       cmp    ax,dx          ; Present >= given?
000031F2 7303                       jae    short atuX     ; Yep! It _was_ inside.
                                atuE:
000031F4 31C9                       xor    cx,cx ; 0      ; Else, fall through with UMB # == -1
000031F6 49                         dec    cx             ; (that makes it return 0xFFFF and sets CF)
                                atuX:
000031F7 89C8                       mov    ax,cx          ; Return the UMB number in AX

000031F9 07                         pop    es
                                    ; 01/01/2023
                                    ;pop   dx
                                    ;pop   cx
000031FA C3                         retn

```

```
                                ; ----------------------------------------------------------------------------
                                ;*** convUMB - checks after GetXNum to convert an address to a UMB number
                                ;             -- if GetXNum read a hex number, we interperete that as a segment
                                ; address rather than a UMB number... and use that address to look up a UMB.
                                ; This routine checks for that condition and calls AddrToUmb if necessary.
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:  AX contains a UMB number or segment, gnradix has been set by GetXNum
                                ; EXIT:   AX will contain a UMB number
                                ; ERROR:  None
                                ; USES:   Flags, AX
                                ; ----------------------------------------------------------------------------

                                    ; 01/01/2023 - Retro DOS v4.2
                                convUMB:
000031FB 2E833E[FE30]10             cmp     word [cs:gnradix],16
00003201 7507                       jne     short cu10      ; If it didn't read in hex, it's not an address
00003203 E8C4FF                     call    AddrToUmb       ; Else, convert the address to a UMB number
                                    ;cmp    ax,0FFFFh
                                    ;jne    short cu10
                                    ;inc    ax              ; If too high, ignore it (make it conventional)
                                    ; 01/01/2023
00003206 40                         inc     ax
00003207 7401                       jz      short cu10      ; If too high, ignore it (make it conventional)
00003209 48                         dec     ax
                                cu10:
0000320A C3                         retn

                                ; 01/01/2023 - Retro DOS v4.2
                                ;%if 0
                                ;
                                ;; ----------------------------------------------------------------------------
                                ;;*** setUMBs - links umbs and sets allocation strategy for a load
                                ;;            -- if LoadHigh, the allocation strategy MAY be LOW_FIRST instead
                                ;; of the usual HIGH_FIRST. See the code.
                                ;; ----------------------------------------------------------------------------
                                ;; ENTRY:  None
                                ;; EXIT:   None
                                ;; ERROR:  None
                                ;; USES:   Flags, fm_umb, fm_strat
                                ;; ----------------------------------------------------------------------------
                                ;
                                ;setUMBs:
                                ;   push    ax
                                ;   push    bx
                                ;   call    fm_link
                                ;   pop     bx
                                ;   pop     ax
                                ;   retn
                                ;
                                ;%endif

                                ; 18/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                ; loadLow subroutine is not used anywhere of IO.SYS 6.22 (& 5.0)
                                %if 0

                                ; ----------------------------------------------------------------------------
                                ;*** loadLow - returns AL==0 if UMB0 == 0, else AL==1
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:  None
                                ; EXIT:   AL==0 if mem strategy should be set to LOW_FIRST, else AL==1
                                ;         Carry set if UMB0 not specified (_NOT_ an error)
                                ; ERROR:  None
                                ; USES:   Flags, fm_strat, fm_umb
                                ; ----------------------------------------------------------------------------
                                ; We want to set the memory strategy to LOW_FIRST if the user specified a
                                ; load UMB, and it is 0. That 0 can be either from the user having _specified_
                                ; zero (/L:0;...), or from having specified a too-big min size (/L:1,99999999)
                                ; such that the load UMB is too small, and shouldn't be used.
                                ; ----------------------------------------------------------------------------

                                loadLow:
                                    ;push   ds
                                    ;push   cs              ; Point DS into appropriate data segment
                                    ;pop    ds

                                    ;mov    al,[UmbLoad]
                                    mov     al,[cs:UmbLoad]
                                    cmp     al,UNSPECIFIED ; 0FFh, -1
                                    jne     short ll10

                                    stc
                                ll15:
                                    mov     al,1            ; Return with AL==1 && STC if no UMBs specified
                                    ;stc
                                    ;jmp    short llX
                                    retn
                                ll10:
                                    or      al,al           ; AL=the load UMB: Is it == 0?
                                    ;jz     short llX       ; Yep... CF==0 (from OR) && AL=0, so just exit

                                    jnz     short ll15      ; 09/04/2019 - Retro DOS v4.0
                                    retn

                                    ;mov    al,1
                                    ;clc
                                ;llX:
                                    ;pop    ds              ; Return DS to where it was
                                    ;retn

                                %endif

                                ; ----------------------------------------------------------------------------
                                ;*** HideUMBs - links UMBs and hides upper-memory as appropriate
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:  None
                                ; EXIT:   None
                                ; ERROR:  None
                                ; USES:   Flags, fm_strat, fm_umb
                                ; ----------------------------------------------------------------------------

                                    ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:322Fh)
                                HideUMBs:
                                    ; 01/01/2023
                                    ;push   ax
                                    ;push   cx
                                    ;push   ds
0000320B 06                         push    es

                                    ; 01/01/2023
                                    ; ds = cs

0000320C E86E02                     call    UmbTest         ; See if we REALLY linked in anything...
0000320F 7232                       jc      short husX      ; ...if not, there's nothing for us to do.
```

```
38405
38406 00003211 E892FD                  call    FixMem          ; Concatenate adjacent free MCBs in upper mem
38407
38408                                  ;call   setUMBs         ; Link UMBs and set memory-allocation strategy
38409                                  ; 01/01/2023
38410 00003214 E8DCFD                  call    fm_link
38411
38412                                  ;putdata fInHigh,1      ; Remember that we're now running high
38413                                  ;mov    byte [cs:fInHigh],1
38414                                  ; 01/01/2023
38415 00003217 C606[FB23]01            mov     byte [fInHigh],1
38416
38417                                  ;call   GetLoadUMB      ; See if they gave us a list to leave free
38418                                  ;mov    al,[cs:UmbLoad] ; 09/04/2019 - Retro DOS v4.0
38419                                  ; 01/01/2023
38420 0000321C A0[FF23]                mov     al,[UmbLoad]
38421
38422 0000321F 3CFF                    cmp     al,UNSPECIFIED ; If they didn't,
38423 00003221 7420                    je      short husX      ; then we shouldn't do this loop:
38424
38425 00003223 31C9                    xor     cx,cx
38426
38427                                ; ------------------------------------------------
38428                                ; HUS10-CX - UMB number (after inc, 1==first UMB)
38429                                ; ------------------------------------------------
38430
38431 00003225 41                   hus10:      inc     cx              ; For each UMB:
38432                                  ; 01/01/2023
38433 00003226 80F910                  cmp     cl,MAXUMB
38434                                  ;cmp    cx,MAXUMB ; 16
38435 00003229 730E                    jae     short hus20
38436
38437 0000322B 88C8                    mov     al,cl           ; (stopping as soon as we're outside of the
38438 0000322D 06                      push    es
38439 0000322E E8A200                  call    findUMB         ; valid range of UMBs)
38440 00003231 07                      pop     es              ; push/pop: trash what findumb finds. :-)
38441 00003232 7205                    jc      short hus20
38442
38443                                  ; 02/01/2023
38444                                  ;push   cx ; *
38445 00003234 E84F01                  call    _hideUMB_       ; hide what we need to hide.
38446                                  ;pop    cx ; *
38447
38448 00003237 EBEC                    jmp     short hus10
38449                                hus20:
38450                                  ;call   GetLoadUMB      ; Now check if they offered /L:0
38451                                  ; 01/01/2023
38452                                  ; ds = cs
38453                                  ;mov    al,[UmbLoad]
38454                                  ;;mov   al,[cs:UmbLoad] ; 09/04/2019 - Retro DOS v4.0
38455 00003239 800E[FF23]00            or      byte [UmbLoad],0
38456                                  ;or     al,al           ; --Is the load UMB 0? (-1==unspecified)
38457 0000323E 7503                    jnz     short husX      ; If not, we're done.
38458
38459 00003240 E86802                  call    hl_unlink       ; If so, however, fix UMBs and strategy.
38460                                husX:
38461 00003243 07                      pop     es
38462                                  ; 01/01/2023
38463                                  ;pop    ds
38464                                  ;pop    cx
38465                                  ;pop    ax
38466 00003244 C3                      retn
38467
38468                                ; ---------------------------------------------------------------------------
38469                                ;*** GetLoadUMB - Returns the load UMB number in AL (-1 if not specified)
38470                                ; ---------------------------------------------------------------------------
38471                                ; ENTRY:  None
38472                                ; EXIT:   AL == load UMB
38473                                ; ERROR:  None
38474                                ; USES:   Flags, AX
38475                                ; ---------------------------------------------------------------------------
38476
38477                                ;GetLoadUMB:
38478                                ;   ;getdata al, UmbLoad
38479                                ;   push    ds
38480                                ;   push    cs
38481                                ;   pop     ds
38482                                ;   mov     al,[UmLoad]
38483                                ;   pop     ds
38484                                ;   retn
38485
38486                                ; ---------------------------------------------------------------------------
38487                                ;*** GetLoadSize - Returns the load UMB minimum size (0 if not specified)
38488                                ; ---------------------------------------------------------------------------
38489                                ; ENTRY:  None
38490                                ; EXIT:   AX == load UMB minimum size
38491                                ; ERROR:  None
38492                                ; USES:   Flags, AX
38493                                ; ---------------------------------------------------------------------------
38494
38495                                ; 08/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
38496                                %if 0
38497                                    ; 01/01/2023 - Retro DOS v4.2
38498                                GetLoadSize:
38499                                    ; 09/04/2019 - Retro DOS v4.0
38500                                    ;mov    al,[cs:UmbLoad]
38501                                    ; 01/01/2023
38502                                    ; ds = cs
38503                                    mov     al,[UmbLoad]
38504                                    ;jmp    short GetSize
38505
38506                                    ;push   bx
38507                                    ;;push  si
38508                                    ;push   ds
38509                                    ;push   cs
38510                                    ;pop    ds
38511
38512                                    ;mov    al,[UmbLoad]
38513
38514                                    ;xor    ah,ah                   ;    ax==UMB
38515                                    ;mov    bx,UmbSize              ;    bx==array
38516                                    ;shl    al,1                     ;    ax==offset
38517                                    ;;add   ax,bx                   ;    ax==element index
38518                                    ;;mov   si,ax                   ; ds:si==element index
38519
38520                                    ;;lodsw                         ;    hh
38521
38522                                    ;add    bx,ax
38523                                    ;mov    ax,[bx]
38524
38525                                    ;pop    ds
38526                                    ;;pop   si
38527                                    ;pop    bx
38528                                    ;retn
```

```
                            %endif

                            ; ----------------------------------------------------------------------------
                            ;*** GetSize - Returns the UMB in AL's minimum size (0 if not specified)
                            ; ----------------------------------------------------------------------------
                            ; ENTRY:  AL == a UMB number
                            ; EXIT:   AX == UMB minimum size, as specified by the user
                            ; ERROR:  None
                            ; USES:   Flags, AX
                            ; ----------------------------------------------------------------------------

                            ; 08/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
                            GetLoadSize:
                                ; ds = cs
                                ;mov   al,[UmbLoad]
                                ; al = [UmbLoad]
                                ; ....

                                ; 01/01/2023 - Retro DOS v4.2
                            GetSize:
                                ; 09/04/2019 - Retro DOS v4.0

                                ;push  bx ; 01/01/2023
                                ;push  si
                                ;push  ds
                                ;push  cs
                                ;pop   ds

00003245 30E4                   xor     ah,ah                   ;    ax==UMB
00003247 BB[1024]               mov     bx,UmbSize              ;    bx==array
0000324A D0E0                   shl     al,1                     ;    ax==offset
                                ;add   ax,bx                   ;    ax==element index
                                ;mov   si,ax                   ; ds:si==element index

                                ;lodsw                          ;    ax==size

0000324C 01C3                   add     bx,ax
                                ; 01/01/2023
                                ; ds = cs
0000324E 8B07                   mov     ax,[bx]
                                ;mov   ax,[cs:bx]

                                ;pop   ds
                                ;pop   si
                                ;pop   bx ; 01/01/2023
                            sls10:      ; 08/09/2023
00003250 C3                     retn

                            ; ----------------------------------------------------------------------------
                            ;*** StoLoadUMB - Overrides the load UMB number with what's in AL
                            ; ----------------------------------------------------------------------------
                            ; ENTRY:   AL == new load UMB
                            ; EXIT:    None
                            ; ERROR:   None
                            ; USES:    Flags, AX
                            ; ----------------------------------------------------------------------------
                            ; CAUTION: Should only be used if /L:... was used.  Logically, that is the only
                            ;          time you would ever need this, so that's okay.
                            ; ----------------------------------------------------------------------------

                            ; StoLoadUMB subroutine is not used anywhere
                            ; of PCDOS 7.1 IBMBIO.COM (& MSDOS 6.21 IO.SYS)
                            ; Erdogan Tan - 18/07/2023

                            ;StoLoadUMB:
                            ;   ;putdata UmbLoad, al
                            ;   push    es
                            ;   push    cs
                            ;   pop     es              ; mov [cs:UmbLoad], al !!!! ; 08/09/2023
                            ;   mov     [es:UmbLoad],al
                            ;   pop     es
                            ;   retn

                            ; ----------------------------------------------------------------------------
                            ;*** StoLoadSize - Overrides the load UMB minimum size with what's in AX
                            ; ----------------------------------------------------------------------------
                            ; ENTRY:  AL == new load size
                            ; EXIT:   None
                            ; ERROR:  None
                            ; USES:   Flags, AX
                            ; ----------------------------------------------------------------------------
                                ; 08/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
                                ; 01/01/2023 - Retro DOS v4.2
                            StoLoadSize:
                                ; 01/01/2023
                                ;push  dx

                                ;getdata dl, UmbLoad          ; Put UMB# in DL and size in AX
                                ;
                                ;push  ds
                                ;push  cs
                                ;pop   ds
                                ;mov   dl,[UmbLoad]
                                ;pop   ds

                                ; 08/09/2023
                                ; MSDOS 6.21 IO.SYS - SYSINIT:32B6h
                                ; PCDOS 7.1 IBMBIO.COM - SYSINIT:3831h

                                ;;mov  dl,[UmbLoad]           ; BUG ! CL would/must be used here
                                                              ; instead of DL (*) ; 18/07/2023
                                ;mov   dl,[cs:UmbLoad] ; Retro DOS v4.0, v4.1, v4.2
                                ;cmp   dl,UNSPECIFIED ; 0FFh
                                ;je    short sls10

                                       ; BUG ! stowSiz uses CL instead of DL !
                                       ; (CL is set in ParseL which calls stowSiz)
                                       ; (This BUG existing in PCDOS 7.1 IBMBIO.COM also)
                                       ; Erdogan Tan - 18/07/2023

                                ; 08/09/2023 (BugFix)
                                ;mov   cl,[cs:UmbLoad]
                                ; 08/09/2023
                                ; ds = cs
00003251 8A0E[FF23]             mov     cl,[UmbLoad]
00003255 80F9FF                 cmp     cl,UNSPECIFIED ; 0FFh
00003258 74F6                   je      short sls10

                                ; 08/09/2023
                            ;   call    stowSiz             ; We've got a function to do just this
                            ;sls10:
                            ;   ; 01/01/2023
                            ;   ;pop   dx
                            ;   retn
```

```
                                    ; 08/09/2023
                                    ;;jmp   stowSiz
                                    ;jmp    short stowSiz

                                ; 08/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
                                %if 1
                                ; ----------------------------------------------------------------------------
                                ;*** stowSiz - marks a given UMB as having a given minimum size
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:    CL contains UMB number, AX contains size
                                ; EXIT:     None
                                ; ERROR:    None
                                ; USES:     AX, DX, Flags, variables in highvar.inc
                                ; ----------------------------------------------------------------------------

                                ; 13/05/2019

                                    ; 01/01/2023 - Retro DOS v4.2
                                stowSiz:
                                    ; 01/01/2023
                                    ;push   bx
                                    ;;push  di ; ?
                                    ;push   es

                                    ;push   cs
                                    ;pop    es

0000325A 88CB                       mov     bl,cl                   ; Now bl==UMB number, AX==size
0000325C B700                       mov     bh,0                    ;     bx==UMB number, AX==size
0000325E D0E3                       shl     bl,1                    ;     bx==offset into array, AX=size
                                    ;mov    [es:bx+UmbSize],ax      ; Store the size
                                    ; 01/01/2023
00003260 2E8987[1024]               mov     [cs:bx+UmbSize],ax      ; Store the size

                                    ; 01/01/2023
                                    ;pop    es
                                    ;;pop   di ; ?
                                    ;pop    bx

00003265 C3                         retn
                                %endif

                                ; ----------------------------------------------------------------------------
                                ;*** hideUMB - marks as HIDDEN all FREE elements in UMB passed as AL
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:    AL must indicate a valid UMB; 0==conv && is invalid.
                                ; EXIT:     None; free elements in UMB marked as hidden
                                ; ERROR:    None
                                ; USES:     Flags
                                ; ----------------------------------------------------------------------------

                                    ; 01/01/2023 - Retro DOS v4.2
                                hideUMB:
                                    ; 02/01/2023
00003266 52                         push    dx ; (*)
                                    ; 01/01/2023
                                    ;push   ax
00003267 06                         push    es

00003268 E86800                     call    findUMB ; (*)   ; Returns with carry if err, else ES == MCB
0000326B 7224                       jc      short huX

                                ; ------------------------------------------------
                                ; HU10--ES - MCB inside UMB; if it's a system MCB,
                                ;            we're not in the same UMB, so exit.
                                ; ------------------------------------------------

0000326D E84AFF                 hu10:       call    isSysMCB        ; Returns with ZF set if owner is SYSTEM
00003270 741F                       jz      short huX       ; If it is, we've finished the UMB.
                                    ;call   isFreeMCB       ; Returns with ZF set if owner is 0
00003272 26830E010000               or      word [es:ARENA.OWNER],0
00003278 7503                       jnz     short hu20

0000327A E81700                     call    hideMCB
                                hu20:
                                    ;mov    al,[es:ARENA.SIGNATURE]
                                    ;cmp    al,arena_signature_end  ;'Z'
                                    ; 19/07/2023
0000327D 26803E00005A               cmp     byte [es:ARENA.SIGNATURE],'Z'
00003283 740C                       jz      short huX       ; 'Z' means this was the last MCB... that's it.

                                    ;NextMCB es,ax          ; Go on forward.
00003285 8CC0                       mov     ax,es
                                    ;add    ax,[es:3]
00003287 2603060300                 add     ax,[es:ARENA.SIZE]
0000328C 40                         inc     ax
0000328D 8EC0                       mov     es,ax

0000328F EBDC                       jmp     short hu10
                                huX:
00003291 07                         pop     es
                                    ; 01/01/2023
                                    ;pop    ax
                                    ; 02/01/2023
00003292 5A                         pop     dx ; (*)
00003293 C3                         retn

                                ; 02/01/2023
                                %if 0

                                ; ----------------------------------------------------------------------------
                                ;*** isTiny - returns with ZF set if user didn't specify /S
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:    None
                                ; EXIT:     ZF set if user DIDN'T specify /S
                                ; ERROR:    None
                                ; USES:     Flags
                                ; ----------------------------------------------------------------------------

                                    ; 02/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                isTiny:
                                    ; 02/01/2023
                                    ;push   ax

                                    ;getdata al,fUmbTiny
                                    ;
                                    ;push   ds
                                    ;push   cs
                                    ;pop    ds
                                    ;mov    al,[fUmbTiny]
                                    ;pop    ds

                                    ; 09/09/2023
```

```
                            ;mov    al,[cs:fUmbTiny]
                            ; 02/01/2023
                            ; ds = cs
                            mov     al,[fUmbTiny]

                            or      al,al
                            ; 02/01/2023
                            ;pop    ax
                            retn

                        %endif

                        ; ----------------------------------------------------------------------------
                        ;*** isFreeMCB - returns with ZF set if current MCB (ES:0) is FREE
                        ; ----------------------------------------------------------------------------
                        ; ENTRY:    ES:0 should point to an MCB
                        ; EXIT:     ZF set if MCB is free, else !ZF
                        ; ERROR:    None
                        ; USES:     Flags
                        ; ----------------------------------------------------------------------------

                        ;isFreeMCB:
                        ;   or      word [es:ARENA.OWNER],0
                        ;   retn

                        ; ----------------------------------------------------------------------------
                        ;*** hideMCB - marks as HIDDEN the MCB at ES:0
                        ; ----------------------------------------------------------------------------
                        ; ENTRY:    ES:0 should point to an MCB
                        ; EXIT:     None; MCB marked as HIDDEN
                        ; ERROR:    None
                        ; USES:     None
                        ; ----------------------------------------------------------------------------

                        hideMCB:
000032A4 26C70601000800     mov     word [es:ARENA.OWNER],SystemPSPOwner ; 8
000032AB 26C70608004849     mov     word [es:ARENA.NAME+0], 'HI' ; 4948h
000032A2 26C7060A004444     mov     word [es:ARENA.NAME+2], 'DD' ; 4444h
000032A9 26C7060C00454E     mov     word [es:ARENA.NAME+4], 'EN' ; 4E45h
000032B0 26C7060E002020     mov     word [es:ARENA.NAME+6], '  ' ; 2020h
000032B7 C3                 retn

                        ; ----------------------------------------------------------------------------
                        ;*** unHideMCB - marks as FREE the MCB at ES:0
                        ; ----------------------------------------------------------------------------
                        ; ENTRY:    ES:0 should point to an MCB
                        ; EXIT:     None; MCB marked as FREE
                        ; ERROR:    None
                        ; USES:     None
                        ; ----------------------------------------------------------------------------

                            ; 03/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)

                        unHideMCB:
                            ; 03/01/2023
                            ;push   ax
000032B8 26C70601000000     mov     word [es:ARENA.OWNER],FreePSPOwner ; 0
000032BF B82020             mov     ax,'  ' ; 2020h
000032C2 26A30800           mov     [es:ARENA.NAME+0],ax
000032C6 26A30A00           mov     [es:ARENA.NAME+2],ax
000032CA 26A30C00           mov     [es:ARENA.NAME+4],ax
000032CE 26A30E00           mov     [es:ARENA.NAME+6],ax
                            ; 03/01/2023
                            ;pop    ax
000032D2 C3                 retn

                        ; ----------------------------------------------------------------------------
                        ;*** findUMB - makes ES:0 point to the first MCB in UMB given as AL
                        ;              -- returns UmbHEAD pointer (0x9FFF) if passed AL==0
                        ; ----------------------------------------------------------------------------
                        ; ENTRY:    AL should be to a valid UMB number
                        ; EXIT:     ES:0 points to first MCB in UMB (_not_ the 8+SC MCB that heads it)
                        ; ERROR:    Carry set if couldn't reach UMB (too high)
                        ; USES:     Flags, ES
                        ; ----------------------------------------------------------------------------

                            ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                            ; (SYSINIT:3344h)
                        findUMB:
                            ; 01/01/2023
                            ;push   ax
                            ; 02/01/2023
000032D3 51                 push    cx ; *
                            ;push   dx

000032D4 30E4               xor     ah,ah           ; Zap ah, so al==ax

000032D6 89C2               mov     dx,ax           ; Store the to-be-found UMB number in DX

000032D8 E8D2FE             call    UmbHead         ; Returns first UMB segment in AX
                            ; 22/07/2023
                            ;mov    es,ax ; *
000032DB 31C9               xor     cx,cx           ; Pretend we're on UMB 0 for now...

                            ; 22/07/2023
                        fu10:
000032DD 8EC0               mov     es,ax ; * ; **
                        ; ------------------------------------------
                        ; FU10--CX - This UMB number; 0 == conventional
                        ;       DX - The UMB number they're looking for
                        ;       ES - The current MCB address
                        ; ------------------------------------------

                        ;fu10:
000032DF 39D1               cmp     cx,dx           ; If CX==DX, we've found the UMB we're
000032E1 7417               je      short fuX       ; searching for--so exit.

000032E3 E8D4FE             call    isSysMCB        ; Returns with ZF set if owner is SYSTEM
000032E6 7501               jnz     short fu20

000032E8 41                 inc     cx              ; If it _was_ SYSTEM, we're in a new UMB.
                        fu20:
                            ;mov    al,[es:ARENA.SIGNATURE]
                            ;cmp    al,arena_signature_end ; 'Z'
                            ; 19/07/2023
000032E9 26803E00005A       cmp     byte [es:ARENA.SIGNATURE],arena_signature_end
000032EF 7408               je      short fuE       ; 'Z' means this was the last MCB... that's it.

                            ;NextMCB es,ax          ; Go on forward.
                            ; 22/07/2023
                            ; ax = es
                            ;mov    ax,es ; * ; 22/07/2023
                            ;add    ax,[es:3]
000032F1 2603060300         add     ax,[es:ARENA.SIZE]
```

```
000032F6 40                             inc     ax
                                        ; 22/07/2023
                                        ;mov    es,ax ; **
000032F7 EBE4                           jmp     short fu10
                                fuE:
000032F9 F9                             stc
                                fuX:
                                        ; 01/01/2023
                                        ;pop    dx
                                        ; 02/01/2023
000032FA 59                             pop     cx ; *
                                        ;pop    ax              ; The address is already in ES.
000032FB C3                             retn

                                ; ---------------------------------------------------------------------------
                                ;*** BigFree - makes ES:0 point to the largest free MCB in UMB given as AL
                                ; ---------------------------------------------------------------------------
                                ; ENTRY:    AL should be to a valid UMB number
                                ; EXIT:     ES:0 points to largest free MCB in UMB, AX returns its size
                                ; ERROR:    Carry set if couldn't reach UMB (0 or too high)
                                ; USES:     Flags, ES
                                ; ---------------------------------------------------------------------------

                                        ; 01/01/2023 - Retro DOS v4.2
                                BigFree:
                                        ; 01/01/2023
                                        ;push   bx
000032FC 51                             push    cx

000032FD E8D3FF                         call    findUMB                 ; Returns with CF if err, else ES==MCB
00003300 723A                           jc      short bfX               ; (would be "jc bfE"; it just does stc)

00003302 31DB                           xor     bx,bx                   ; Segment address of largest free MCB
00003304 31C9                           xor     cx,cx                   ; Size of largest free MCB

                                ; ---------------------------------------------
                                ; BF10--ES - Current MCB address
                                ;       BX - Address of largest free MCB so far
                                ;       CX - Size of largest free MCB so far
                                ; ---------------------------------------------

                                bf10:
00003306 E8B1FE                         call    isSysMCB                ; If we've left the MCB, we're done.
00003309 7428                           jz      short bf30

                                        ;call   isFreeMCB               ; Returns with ZF set if owner is 0
0000330B 26830E010000                   or      word [es:ARENA.OWNER],0
00003311 750C                           jnz     short bf20

00003313 26A10300                       mov     ax,[es:ARENA.SIZE]
                                        ;cmp    cx,[es:ARENA.SIZE]      ; Compare sizes...
00003317 39C1                           cmp     cx,ax
                                        ;jg     short bf20              ; Unless we're bigger,
                                        ; 19/07/2023
00003319 7D04                           jge     short bf20

0000331B 8CC3                           mov     bx,es                   ; Store this new element's address,
                                        ;mov    cx,[es:ARENA.SIZE]      ; and its size.
0000331D 89C1                           mov     cx,ax

                                bf20:
                                        ;mov    al,[es:ARENA.SIGNATURE]
                                        ;cmp    al,arena_signature_end; 'Z'
                                        ; 19/07/2023
                                        ;cmp    byte [es:0],'Z'
0000331F 26803E00005A                   cmp     byte [es:ARENA.SIGNATURE],arena_signature_end
00003325 740C                           jz      short bf30              ; 'Z' means this was the last MCB.

                                        ;NextMCB es,ax                  ; Go on forward.
00003327 8CC0                           mov     ax,es
                                        ;add    ax,[es:3]
00003329 2603060300                     add     ax,[es:ARENA.SIZE]
0000332E 40                             inc     ax
0000332F 8EC0                           mov     es,ax

00003331 EBD3                           jmp     short bf10

00003333 8EC3                   bf30:   mov     es,bx                   ; Return the address
00003335 89C8                           mov     ax,cx                   ; Return the size
00003337 09DB                           or      bx,bx
00003339 7501                           jnz     short bfX               ; (if size==0, there's nothing free)
                                bfE:
0000333B F9                             stc
                                bfX:
0000333C 59                             pop     cx
                                        ; 01/01/2023
                                        ;pop    bx
0000333D C3                             retn

                                ; ---------------------------------------------------------------------------
                                ;*** isSpecified - sets ZF if UMB in AL wasn't specified in DH/LH line.
                                ; ---------------------------------------------------------------------------
                                ; ENTRY:    AL should be to a valid UMB number
                                ; EXIT:     ZF set if UMB wasn't specified, ZF clear if it was
                                ; ERROR:    None
                                ; USES:     Flags
                                ; ---------------------------------------------------------------------------

                                        ; 02/01/2023 - Retro DOS v4.2

                                isSpecified:
                                        ; 02/01/2023
                                        ;push   ax

0000333E 30FF                           xor     bh,bh
00003340 88C3                           mov     bl,al

                                        ;getdata al,DS:UmbUsed[bx]
                                        ;
                                        ;push   ds
                                        ;push   cs
                                        ;pop    ds
                                        ;mov    al,[bx+UmbUsed]
                                        ;pop    ds

                                        ;mov    al,[cs:bx+UmbUsed]
                                        ; 02/01/2023
                                        ; ds = cs
00003342 8A87[0024]                     mov     al,[bx+UmbUsed]

00003346 08C0                           or      al,al                   ; Sets ZF if al==0 (ie, if unspecified)

                                        ; 09/09/2023
                                        ; 02/01/2023
```

```
                                    ;pop    ax
                                    retn

                                ; ------------------------------------------------------------------------------
                                ;*** shrinkMCB - breaks an MCB into two pieces, the lowest one's size==AX
                                ; ------------------------------------------------------------------------------
                                ; ENTRY:    AX == new size, ES:0 == current MCB
                                ; EXIT:     None; MCB broken if carry clear
                                ; ERROR:    Carry set if MCB isn't as large as AX+0x20 (not a useful split)
                                ; USES:     Flags
                                ; ------------------------------------------------------------------------------
                                ; If the size of the to-be-split MCB isn't at least 0x20 bytes greater than
                                ; the specified new size, the split is useless; if it's only 0x10 bytes, that
                                ; 0x10 will be used to make a header that mentions a 0-byte free space, and
                                ; that just sucks up 0x10 bytes for nothing. So we make 0x20 bytes the
                                ; minimum for performing a split.
                                ; ------------------------------------------------------------------------------

                                MIN_SPLIT_SIZE     equ     20h

                                    ; 02/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)

                                shrinkMCB:
                                    ;pushreg <bx,cx,es>
                                    ; 02/01/2023
                                    ;push   bx
00003349 51                         push    cx
0000334A 06                         push    es

0000334B 89C3                       mov     bx,ax                   ; Move things around... and
                                    ; 02/01/2023
                                    ;mov    ax,es                   ; save this one for later.

0000334D 268B0E0300                 mov     cx,[es:ARENA.SIZE]
                                    ; 02/01/2023
00003352 89C8                       mov     ax,cx

00003354 83E820                     sub     ax,MIN_SPLIT_SIZE ; 32
                                    ;sub    cx,MIN_SPLIT_SIZE ; 32
                                    ;;cmp   bx,cx                   ; {New size} vs {Current Size-20h}
                                    ;ja     short smE               ; if wanted_size > cur-20h, abort.
                                    ; 18/12/2022
                                    ;cmp    cx,bx
                                    ; 02/01/2023
00003357 39D8                       cmp     ax,bx
00003359 7228                       jb      short smE ; (*)

0000335B 268A160000                 mov     dl,[es:ARENA.SIGNATURE]

                                    ;mov    cx,[es:ARENA.SIZE]
                                    ; 02/01/2023
00003360 8CC0                       mov     ax,es

00003362 26891E0300                 mov     [es:ARENA.SIZE],bx
00003367 26C60600004D               mov     byte [es:ARENA.SIGNATURE],'M'

0000336D 01D8                       add     ax,bx
0000336F 40                         inc     ax
00003370 8EC0                       mov     es,ax                   ; Move to new arena area

00003372 89C8                       mov     ax,cx
00003374 29D8                       sub     ax,bx
                                    ; 12/12/2022
                                    ; ax > 0
00003376 48                         dec     ax                      ; And prepare the new size

                                    ; 18/12/2022
00003377 2688160000                 mov     [es:ARENA.SIGNATURE],dl
                                    ;mov    word [es:ARENA.OWNER],0 ; (**)
0000337C 26A30300                   mov     [es:ARENA.SIZE],ax
                                    ;mov    ax,'  ' ; 2020h
                                    ;mov    [es:ARENA.NAME+0],ax ; (**)
                                    ;mov    [es:ARENA.NAME+2],ax ; (**)
                                    ;mov    [es:ARENA.NAME+4],ax ; (**)
                                    ;mov    [es:ARENA.NAME+6],ax ; (**)

                                    ; 18/12/2022
00003380 E8A801                     call    freeMCB ; (**)

                                    ; 12/12/2022
                                    ; cf=0
                                    ;clc
                                    ; 18/12/2022
                                    ;jmp    short smX
                                smE:
                                    ; 18/12/2022
                                    ; cf=1 (*)
                                    ;stc
                                smX:
                                    ;popreg <es,cx,bx>
00003383 07                         pop     es
00003384 59                         pop     cx
                                    ; 02/01/2023
                                    ;pop    bx
00003385 C3                         retn

                                ; ------------------------------------------------------------------------------
                                ;*** hideUMB? - hides as appropriate the UMB in CL
                                ; ------------------------------------------------------------------------------
                                ; ENTRY:    CL should be to a valid UMB number, and AX to its address (findUMB)
                                ; EXIT:     None; UMB is hidden as necessary
                                ; ERROR:    None
                                ; USES:     Flags, AX, CX
                                ; ------------------------------------------------------------------------------
                                ; PRIMARY LOGIC:
                                ;
                                ; If the UMB is specified in the DH/LH statement, then:
                                ;    If the largest free segment is too small (check specified size), then:
                                ;       Pretend it wasn't ever specified, and fall out of this IF.
                                ;    Else, if largest free segment is LARGER than specified size, then:
                                ;       If /S was given on the command-line, then:
                                ;          Break that element into two pieces
                                ;          Set a flag that we're shrinking
                                ;       Endif
                                ;    Endif
                                ; Endif
                                ; If the UMB is NOT specified (or was removed by the above):
                                ;    Hide all free elements in the UMB
                                ;    If the flag that we're shrinking was set, then:
                                ;       UN-hide the lower portion of the shrunken UMB
                                ;    ENDIF
                                ; ENDIF
                                ; ------------------------------------------------------------------------------
```

```
                                    ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:3426h)
                                _hideUMB_:
                                    ; 02/01/2023
                                    ; ds = cs

                                    ; 01/01/2023
                                    ;push  bx
                                    ;push  dx
00003386 06                         push   es

00003387 88C8                       mov    al,cl
00003389 E8B2FF                     call   isSpecified    ; Returns ZF set if al's umb was NOT specified
0000338C 742D                       jz     short hu_20

0000338E 88C8                       mov    al,cl          ; Retrieve the size of the largest
00003390 E869FF                     call   BigFree        ; free element in AX; put its address in ES
00003393 7226                       jc     short hu_20    ; Oops. Errors mean skip this part.

00003395 50                         push   ax             ; TOS==size of BigFree in UMB (popped as BX)
00003396 88C8                       mov    al,cl          ; Retrieve the user's specified
00003398 E8AAFE                     call   GetSize        ; minimum size for this umb (into AX)
0000339B 5B                         pop    bx             ; Now BX==BigFree, AX==Specified Size

0000339C 09C0                       or     ax,ax          ; If they didn't specify one,
0000339E 741B                       jz     short hu_20    ; Skip over all this.

000033A0 39D8                       cmp    ax,bx          ; Ah... if (specified > max free)
000033A2 7607                       jbe    short hu_10

000033A4 88C8                       mov    al,cl          ;  Then mark that UMB as unused. Nya nya.
000033A6 E81DFD                     call   unMarkUMB
000033A9 EB10                       jmp    short hu_20
                                hu_10:
                                    ;call  isTiny         ; Returns ZF clear if user specified /S
                                    ;jz    short hu_20
                                    ; 02/01/2023
                                ;isTiny:
                                    ;mov   al,[fUmbTiny] ; ds = cs
                                    ;or    al,al
000033AB 800E[FC23]00               or     byte [fUmbTiny],0
000033B0 7409                       jz     short hu_20

000033B2 E894FF                     call   shrinkMCB      ; They specified /S, so shrink the MCB to AX
000033B5 7204                       jc     short hu_20    ; Ah... if didn't shrink after all, skip this:

000033B7 8CC2                       mov    dx,es
000033B9 EB09                       jmp    short hu_30    ; Skip the spec check.. we wanna hide this one.

000033BB 89C8                   hu_20:      mov    ax,cx
000033BD E87EFF                     call   isSpecified    ; If they specified this UMB, we're done...
000033C0 7510                       jnz    short hu_X     ; so leave.

000033C2 31D2                       xor    dx,dx
                                hu_30:
000033C4 88C8                       mov    al,cl

000033C6 E89DFE                     call   hideUMB        ; Hides everything in UMB #al

000033C9 09D2                       or     dx,dx          ; Did we shrink a UMB? If not, DX==0,
000033CB 7405                       jz     short hu_X     ; So we should leave.

000033CD 8EC2                       mov    es,dx          ; Ah, but if it isn't, DX==the MCB's address;
000033CF E8E6FE                     call   unHideMCB      ; Un-hides the lower portion of that MCB.
                                hu_X:
000033D2 07                         pop    es
                                    ; 01/01/2023
                                    ;pop   dx
                                    ;pop   bx
000033D3 C3                         retn

                                ; ---------------------------------------------------------------------------
                                ;*** UnFreeze - Marks FROZEN elements as FREE
                                ; ---------------------------------------------------------------------------
                                ; Entry:  None
                                ; Exit:   None; all 8+FROZEN elements are marked as FREE, from any UMB.
                                ; Error:  None
                                ; Uses:   Flags
                                ; ---------------------------------------------------------------------------

                                    ; 03/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                UnFreeze:
                                    ; 03/01/2023
                                    ;push  ax
000033D4 06                         push   es

000033D5 E8D5FD                     call   UmbHead        ; Returns with carry if err, else ES == MCB
000033D8 721C                       jc     short ufX

                                    ; 22/07/2023
                                uf10:
000033DA 8EC0                       mov    es,ax ; *

                                ; ------------------------------
                                ; UF10--ES - Current MCB address
                                ; ------------------------------

                                ;uf10:
000033DC E81900                     call   isFrozMCB      ; Returns with ZF set if MCB is FROZEN
000033DF 7505                       jnz    short uf20
000033E1 E8D4FE                     call   unHideMCB
                                    ; 09/09/2023
                                    ; ax <> es
000033E4 8CC0                       mov    ax,es ; *
                                uf20:
                                    ;mov   al,[es:ARENA.SIGNATURE]
                                    ;cmp   al,arena_signature_end ; 'Z'
                                    ; 22/07/2023
000033E6 26803E00005A               cmp    byte [es:ARENA.SIGNATURE],arena_signature_end ; 'Z'
000033EC 7408                       jz     short ufX      ; 'Z' means this was the last MCB.. that's it.

                                    ;NextMCB es,ax         ; Go on forward.
                                    ; 22/07/2023
                                    ; ax = es
                                    ;mov   ax,es ; *
                                    ;add   ax,[es:3]
000033EE 2603060300                 add    ax,[es:ARENA.SIZE]
000033F3 40                         inc    ax
                                    ; 22/07/2023
                                    ;mov   es,ax
000033F4 EBE4                       jmp    short uf10
                                ufX:
000033F6 07                         pop    es
```

```
                                    ; 03/01/2023
                                    ;pop    ax
000033F7 C3                         retn

                                ; ---------------------------------------------------------------------------
                                ;*** isFrozMCB - returns with ZF set if current MCB (ES:0) is FROZEN
                                ; ---------------------------------------------------------------------------
                                ; ENTRY:    ES:0 should point to an MCB
                                ; EXIT:     ZF set if MCB is frozen, else !ZF
                                ; ERROR:    None
                                ; USES:     Flags
                                ; ---------------------------------------------------------------------------

                                isFrozMCB:
                                    ;push   ax

                                    ;mov    ax,[es:ARENA.OWNER]    ; Check the owner...
                                    ;cmp    ax,SystemPSPOwner      ; 8 (for US OR Japan) is valid
000033F8 26833E010008               cmp     word [es:ARENA.OWNER],SystemPSPOwner
000033FE 7522                       jne     short ifmX

                                    ;mov    ax,[es:ARENA.NAME+0]
                                    ;cmp    ax,'FR' ; 5246h
00003400 26813E08004652             cmp     word [es:ARENA.NAME+0],'FR'
00003407 7519                       jne     short ifmX
                                    ;mov    ax,[es:ARENA.NAME+2]
                                    ;cmp    ax,'OZ' ; 5A4Fh
00003409 26813E0A004F5A             cmp     word [es:ARENA.NAME+2],'OZ'
00003410 7510                       jne     short ifmX
                                    ;mov    ax,[es:ARENA.NAME+4]
                                    ;cmp    ax,'EN' ; 4E45h
00003412 26813E0C00454E             cmp     word [es:ARENA.NAME+4],'EN'
00003419 7507                       jne     short ifmX
                                    ;mov    ax,[es:ARENA.NAME+6]
                                    ;cmp    ax,'  ' ; 2020h
0000341B 26813E0E002020             cmp     word [es:ARENA.NAME+6],'  '
                                ifmX:
                                    ;pop    ax
00003422 C3                         retn

                                ; ---------------------------------------------------------------------------
                                ;*** frezMCB - marks as 8+FROZEN the MCB at ES:0
                                ; ---------------------------------------------------------------------------
                                ; ENTRY:    ES:0 should point to an MCB
                                ; EXIT:     None; MCB frozen
                                ; ERROR:    None
                                ; USES:     None
                                ; ---------------------------------------------------------------------------

                                frezMCB:
00003423 26C70601000800             mov     word [es:ARENA.OWNER],SystemPSPOwner ; 8
0000342A 26C70608004652             mov     word [es:ARENA.NAME+0],'FR'
00003431 26C7060A004F5A             mov     word [es:ARENA.NAME+2],'OZ'
00003438 26C7060C00454E             mov     word [es:ARENA.NAME+4],'EN'
0000343F 26C7060E002020             mov     word [es:ARENA.NAME+6],'  '
00003446 C3                         retn

                                ; ---------------------------------------------------------------------------
                                ;*** FreezeUM - Marks FROZEN all UM elements now FREE, save those in load UMB
                                ; ---------------------------------------------------------------------------
                                ; Entry:  None
                                ; Exit:   None; all free elements not in load UMB marked as 8+FROZEN
                                ; Error:  None
                                ; Uses:   Flags
                                ; ---------------------------------------------------------------------------

                                    ; 01/01/2023 - Retro DOS v4.2
                                FreezeUM:
                                    ; 01/01/2023
                                    ;push   ax
                                    ;push   cx
                                    ;push   dx
00003447 06                         push    es

                                    ;;call  GetLoadUMB
                                    ; 01/01/2023
                                    ; ds = cs
                                    ;mov    al,[cs:UmbLoad] ; 19/04/2019 - Retro DOS v4.0
00003448 A0[FF23]                   mov     al,[UmbLoad]

0000344B 30E4                       xor     ah,ah           ; Zap ah, so al==ax
0000344D 89C2                       mov     dx,ax           ; Store the load UMB in DX, so we can skip it

0000344F E85BFD                     call    UmbHead         ; Returns first UMB segment in AX
                                    ; 22/07/2023
                                    ;mov    es,ax ; *
00003452 31C9                       xor     cx,cx           ; Pretend we're on UMB 0 for now...

                                    ; 22/07/2023
                                fum10:
00003454 8EC0                       mov     es,ax ; *

                                ; ----------------------------------------
                                ; FUM10--ES - Current MCB address
                                ;        CX - Current UMB number
                                ;        DX - UMB number to skip (load UMB)
                                ; ----------------------------------------

                                ;fum10:
00003456 E861FD                     call    isSysMCB        ; Returns with ZF set if owner is SYSTEM
00003459 7501                       jnz     short fum20

0000345B 41                         inc     cx              ; If it _was_ SYSTEM, we're in a new UMB.
                                fum20:
0000345C 39D1                       cmp     cx,dx           ; If this is the load UMB, we don't want to
0000345E 740B                       je      short fum30     ; freeze anything... so skip that section.

                                    ;call   isFreeMCB       ; Oh. If it's not free, we can't freeze it
00003460 26830E010000               or      word [es:ARENA.OWNER],0
00003466 7503                       jnz     short fum30     ; either.

00003468 E8B8FF                     call    frezMCB
                                fum30:
                                    ;mov    al,[es:ARENA.SIGNATURE]
                                    ;cmp    al,arena_signature_end ; 'Z'
                                    ; 22/07/2023
0000346B 26803E00005A               cmp     byte [es:ARENA.SIGNATURE],arena_signature_end ; 'Z'
00003471 7408                       je      short fumX      ; 'Z' means this was the last MCB.. that's it.

                                    ;NextMCB es, ax                ; Go on forward.
                                    ; 22/07/2023
                                    ; ax = es
                                    ;mov    ax,es
                                    ;add    ax,[es:3]
```

```
39397 00003473 2603060300                add     ax,[es:ARENA.SIZE]
39398 00003478 40                        inc     ax
39399                                    ; 22/07/2023
39400                                    ;mov    es,ax ; *
39401 00003479 EBD9                      jmp     short fum10
39402
39403 0000347B 07                      fumX:     pop     es
39404                                    ; 01/01/2023
39405                                    ;pop    dx
39406                                    ;pop    cx
39407                                    ;pop    ax
39408 0000347C C3                        retn
39409
39410                                  ; ------------------------------------------------------------------------------
39411                                  ;*** UmbTest - returns with carry set if UMBs are not available, else CF==false
39412                                  ; ------------------------------------------------------------------------------
39413                                  ; ENTRY:    None
39414                                  ; EXIT:     Carry is clear if UMBs are available, or set if they are not
39415                                  ; ERROR:    None
39416                                  ; USES:     CF (AX,BX,DS,ES pushed 'cause they're used by others)
39417                                  ; ------------------------------------------------------------------------------
39418
39419                                      ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
39420                                  UmbTest:
39421                                      ; 01/01/2023
39422                                      ;push   ax
39423 0000347D 53                        push    bx ; *
39424                                    ;push   ds
39425 0000347E 06                        push    es ; **
39426
39427                                    ; 01/01/2023
39428                                    ; ds = cs
39429
39430 0000347F E871FB                    call    fm_link                 ; Link in UMBs (if not already linked)
39431 00003482 E80800                    call    WalkMem                 ; Check to see if they're really linked
39432 00003485 9C                        pushf                           ; And remember what we found out
39433 00003486 E87BFB                    call    fm_unlink               ; Unlink UMBs (if WE have linked 'em)
39434 00003489 9D                        popf                            ; And restore what we found out.
39435
39436 0000348A 07                        pop     es ; **
39437                                    ; 01/01/2023
39438                                    ;pop    ds
39439 0000348B 5B                        pop     bx ; *
39440                                    ;pop    ax
39441 0000348C C3                        retn
39442
39443                                  ; ------------------------------------------------------------------------------
39444                                  ;*** WalkMem - travels memory chain and returns carry clear iff UMBs are linked
39445                                  ; ------------------------------------------------------------------------------
39446                                  ; ENTRY:    None
39447                                  ; EXIT:     Carry SET if MCB chain stops before 9FFF, CLEAR if stops >= 9FFF.
39448                                  ; ERROR:    None
39449                                  ; USES:     Flags
39450                                  ; ------------------------------------------------------------------------------
39451
39452                                      ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
39453                                      ; (SYSINIT:3541h)
39454
39455                                  WalkMem:
39456                                      ;push   ax ; ?
39457                                      ;push   bx ; ?
39458                                      ;;push  ds ; ? ; 01/01/2023 (MSDOS 6.21 IO.SYS, SYSINIT:352Fh)
39459                                      ;push   es ; ? no need to save contents of these registers ?
39460
39461 0000348D B452                      mov     ah,GET_IN_VARS          ; Call int 21h, function 52h...
39462 0000348F CD21                      int     21h
39463
39464 00003491 268B47FE                  mov     ax,[es:bx-2]
39465                                    ; 22/07/2023
39466                                  um10:
39467 00003495 8EC0                      mov     es,ax ; * ; **
39468
39469                                  ; ------------------------------
39470                                  ; UM10: ES = Current MCB pointer
39471                                  ; ------------------------------
39472
39473                                  ;um10:
39474                                      ;mov    al,[es:ARENA.SIGNATURE]
39475                                      ;cmp    al,arena_signature_end ; 'Z'
39476                                      ; 22/07/2023
39477 00003497 26803E00005A              cmp     byte [es:ARENA.SIGNATURE],arena_signature_end ; 'Z'
39478 0000349D 7408                      je      short um20              ; If signature == 'Z', hay no more.
39479
39480                                      ;NextMCB es,bx                  ; Move to the next MCB
39481
39482                                      ;mov    bx,es
39483                                      ;;add   bx,[es:3]
39484                                      ;add    bx,[es:ARENA.SIZE]
39485                                      ;inc    bx
39486                                      ;mov    es,bx
39487                                      ; 22/07/2023
39488                                      ; ax = es
39489                                      ;mov    ax,es ; *
39490 0000349F 2603060300                add     ax,[es:ARENA.SIZE]
39491 000034A4 40                        inc     ax
39492                                      ;mov    es,ax ; **
39493
39494 000034A5 EBEE                      jmp     short um10              ; And restart the loop.
39495                                  um20:
39496                                      ; 22/07/2023
39497                                      ; ax = es
39498                                      ;mov    ax,es
39499
39500 000034A7 3DFF9F                    cmp     ax,9FFFh                ; This sets CF if ax < 9FFF.
39501
39502                                      ;pop    es ; ?
39503                                      ;;pop   ds ; ? ; 01/01/2023 (MSDOS 6.21 IO.SYS, SYSINIT:353Dh)
39504                                      ;pop    bx ; ?
39505                                      ;pop    ax ; ?
39506
39507 000034AA C3                        retn
39508
39509                                  ; ------------------------------------------------------------------------------
39510                                  ;*** hl_unlink - unlinks UMBs if fm_umb is set to 0; restores strategy too
39511                                  ; ------------------------------------------------------------------------------
39512                                  ; ENTRY:    fm_umb == 1 : leave linked, else unlink
39513                                  ; EXIT:     None
39514                                  ; ERROR:    None
39515                                  ; USES:     AX, BX
39516                                  ; ------------------------------------------------------------------------------
39517
39518                                      ; 01/01/2023 - Retro DOS v4.2
39519                                  hl_unlink:
39520 000034AB 30FF                      xor     bh,bh
```

```
                                    ;getdata bl,fm_umb            ; Restore original link-state
                                    ;
                                    ;push   ds
                                    ;push   cs
                                    ;pop    ds
                                    ;mov    bl,[fm_umb]
                                    ;pop    ds

                                    ; 01/01/2023
                                    ; ds = cs
                                    ;mov    bl,[cs:fm_umb]
000034AD 8A1E[3024]                 mov     bl,[fm_umb]

000034B1 B80358                     mov     ax,DOS_SET_UMBLINK ; 5803h
000034B4 CD21                       int     21h
000034B6 C3                         retn

                                ; ----------------------------------------------------------------------
                                ; HIGHEXIT.INC (MSDOS 6.0 - 1991)
                                ; ----------------------------------------------------------------------
                                ; 09/04/2019 - Retro DOS v4.0

                                ;   Module:   HIGHEXIT.INC - Code executed after LoadHigh or DeviceHigh
                                ;   Date:     May 14, 1992

                                ;   Modification log:
                                ;
                                ;     DATE    WHO      DESCRIPTION
                                ;   --------  -------  ------------------------------------------------------
                                ;   05/14/92  t-richj  Original
                                ;   06/21/92  t-richj  Final revisions before check-in

                               UMB_HeadIdx equ     8Ch     ; Offset from ES (after func52h) to get UMBHead

                                ; ----------------------------------------------------------------------------
                                ;*** UnHideUMBs - Marks HIDDEN elements as FREE
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:  None; perhaps, earlier, HideUMBs was called... if not, we have
                                ;              very little to do, as no elelments will be marked as HIDDEN.
                                ; EXIT:   Sets InHigh to zero; carry clear if HideUMBs was called earlier.
                                ; ERROR:  None
                                ; USES:   fInHigh (from highvar.inc), carry flag
                                ; ----------------------------------------------------------------------------

                                    ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:357Bh)

                               UnHideUMBs:
000034B7 50                         push    ax              ; Save ax for what we're about to do

                                ; ----------------------------------------------------------------------------
                                ; BUGBUG t-richj 11-8-92: The following six lines were commented out for a good
                                ;    length of time. Those six constitute a check of whether or not we should
                                ;    indeed clean up the upper-memory chain; without such a check, COMMAND.COM
                                ;    will destroy the current link-state and memory-allocation strategy after
                                ;    every command execution.
                                ; ----------------------------------------------------------------------------

                                    ;getdata al,fInHigh   ; Get InHigh from data segment
                                    ;
                                    ;push   ds
                                    ;push   cs
                                    ;pop    ds
                                    ;mov    al,[fInHigh]
                                    ;pop    ds

                                    ;mov    al,[cs:fInHigh]
                                    ; 31/12/2022
                                    ; ds = cs
000034B8 A0[FB23]                   mov     al,[fInHigh]

000034BB 08C0                       or      al,al
000034BD 7503                       jnz     short uhu10     ; If didn't call loadhigh/devicehigh earlier,

000034BF 58                         pop     ax              ; then there's nothing to do here... so
000034C0 F9                         stc                     ; restore everything and return. Just like
000034C1 C3                         retn                    ; that.
                               uhu10:
000034C2 E88E00                     call    linkumb         ; Make sure UMBs are linked in.
000034C5 E81200                     call    FreeUMBs

                                    ;putdata fInHigh,0    ; We're leaving, so update fInHigh.
                                    ;
                                    ;push   es
                                    ;push   cs
                                    ;pop    es
                                    ;mov    byte [es:fInHigh],0
                                    ;pop    ds

                                    ; 31/12/2022
                                    ; ds = cs
                                    ;mov    byte [cs:fInHigh],0
000034C8 C606[FB23]00               mov     byte [fInHigh],0

                                    ;call   he_unlink       ; Unlink UMBs
                                    ; 31/12/2022
                                ;;he_unlink:
000034CD 30FF                       xor     bh,bh

                                    ;getdata bl,fm_umb     ; Restore original link-state
                                    ;mov    bl,[cs:fm_umb]
000034CF 8A1E[3024]                 mov     bl,[fm_umb]

000034D3 B80358                     mov     ax,DOS_SET_UMBLINK ; 5803h
000034D6 CD21                       int     21h
                                    ;;retn

000034D8 58                         pop     ax
                                    ; 12/12/2022
                                    ;clc    ; 12/12/2022 (this clc may not be necessary!?)
000034D9 C3                         retn

                                ; 31/12/2022
                                ;%if 0
                                ;
                                ;; ----------------------------------------------------------------------------
                                ;;*** he_unlink - unlinks UMBs if fm_umb is set to 0
                                ;; ----------------------------------------------------------------------------
                                ;; ENTRY:    fm_umb == 1 : leave linked, else unlink
                                ;; EXIT:     None
                                ;; ERROR:    None
                                ;; USES:     AX, BX
                                ;; ----------------------------------------------------------------------------
```

```
                                ;
                                ;he_unlink:
                                ;   xor     bh, bh
                                ;
                                ;   ;getdata bl, fm_umb   ; Restore original link-state
                                ;   mov     bl,[cs:fm_umb]
                                ;
                                ;   mov     ax,DOS_SET_UMBLINK ; 5803h
                                ;   int     21h
                                ;   retn
                                ;
                                ;%endif

                                ; ----------------------------------------------------------------------------
                                ;*** freeUMBs - frees all HIDDEN memory elements in upper-memory.
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:    None
                                ; EXIT:     None; HIDDEN memory elements returned to FREE
                                ; ERROR:    None (ignore CF)
                                ; USES:     Flags
                                ; ----------------------------------------------------------------------------

                                FreeUMBs:
000034DA 50                         push    ax
000034DB 06                         push    es

000034DC E86700                     call    HeadUmb         ; Returns with carry if err, else ES == MCB
000034DF 721C                       jc      short fusX
                                fus10:
000034E1 8EC0                       mov     es,ax           ; Prepare for the loop; ES = current MCB addr.
                                ;fus10:
000034E3 E81A00                     call    isHideMCB       ; Returns with ZF set if owner is 0
000034E6 7505                       jnz     short fus20
000034E8 E84000                     call    freeMCB
                                    ; 09/09/2023
                                    ; ax <> es
000034EB 8CC0                       mov     ax,es
                                fus20:
                                    ;mov    al,[es:ARENA.SIGNATURE]
                                    ;cmp    al,arena_signature_end ; 'Z'
                                    ; 22/07/2023
000034ED 26803E00005A               cmp     byte [es:ARENA.SIGNATURE],arena_signature_end ; 'Z'
000034F3 7408                       jz      short fusX      ; That means this was the last MCB--that's it.

                                    ; 22/07/2023
                                    ; ax = es
                                    ;mov    ax,es
000034F5 2603060300                 add     ax,[es:ARENA.SIZE]
000034FA 40                         inc     ax
                                    ; 22/07/2023
                                    ;mov    es,ax
000034FB EBE4                       jmp     short fus10     ; Go on forward.
                                fusX:
000034FD 07                         pop     es
000034FE 58                         pop     ax
000034FF C3                         retn

                                ; ----------------------------------------------------------------------------
                                ;*** isHideMCB - returns with ZF set if current MCB (ES:0) is HIDDEN
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:    ES:0 should point to an MCB
                                ; EXIT:     ZF set if MCB is hidden, else !ZF
                                ; ERROR:    None
                                ; USES:     Flags
                                ; ----------------------------------------------------------------------------

                                isHideMCB:
                                    ;push   ax

00003500 26833E010008               cmp     word [es:ARENA.OWNER],SystemPSPOwner ; If the owner's SYSTEM
00003506 7522                       jne     short ihm_x                          ; then check for HIDDEN

                                    ;mov    ax,[es:ARENA.NAME]
                                    ;cmp    ax,'HI' ; 4948h
00003508 26813E08004849             cmp     word [es:ARENA.NAME+0],'HI'
0000350F 7519                       jne     short ihm_x
                                    ;mov    ax,[es:ARENA.NAME+2]
                                    ;cmp    ax,'DD' ; 4444h
00003511 26813E0A004444             cmp     word [es:ARENA.NAME+2],'DD'
00003518 7510                       jne     short ihm_x
                                    ;mov    ax,[es:ARENA.NAME+4]
                                    ;cmp    ax,'EN' ; 4E45h
0000351A 26813E0C00454E             cmp     word [es:ARENA.NAME+4],'EN'
00003521 7507                       jne     short ihm_x
                                    ;mov    ax,[es:ARENA.NAME+6]
                                    ;cmp    ax,'  ' ; 2020h
00003523 26813E0E002020             cmp     word [es:ARENA.NAME+6],'  '
                                ihm_x:
                                    ;pop    ax
0000352A C3                         retn

                                ; ----------------------------------------------------------------------------
                                ;*** freeMCB - marks as free the MCB at ES:0
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:    ES:0 should point to an MCB
                                ; EXIT:     None; MCB free'd
                                ; ERROR:    None
                                ; USES:     AX
                                ; ----------------------------------------------------------------------------

                                freeMCB:
0000352B 26C70601000000             mov     word [es:ARENA.OWNER],0
00003532 B82020                     mov     ax,'  ' ; mov ax,2020h ; 31/12/2022
00003535 26A30800                   mov     [es:ARENA.NAME+0],ax
00003539 26A30A00                   mov     [es:ARENA.NAME+2],ax
0000353D 26A30C00                   mov     [es:ARENA.NAME+4],ax
00003541 26A30E00                   mov     [es:ARENA.NAME+6],ax
00003545 C3                         retn

                                ; ----------------------------------------------------------------------------
                                ;*** HeadUmb - returns in AX the address of the first UMB block (0x9FFF)
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:  Nothing
                                ; EXIT:   AX contains 0x9FFF for most systems
                                ; ERROR:  Carry set if pointer is 0xFFFF (if not set up yet--DH runs into this)
                                ; USES:   Flags, AX
                                ; ----------------------------------------------------------------------------

                                HeadUmb:
                                    ; 13/05/2019

                                    ;push   si ; ?
                                    ;push   ds ; ?
                                    ;push   es
```

```
                                    ;push   bx ; *
                                    ; 09/04/2019
                                    ; !!! No need to save es,bx,ds,si above !!! (es,bx are changed here)
00003546 B452                       mov     ah,GET_IN_VARS          ; Call int 21h, function 52h...
00003548 CD21                       int     21h
                                            ; DOS - 2+ internal - GET LIST OF LISTS
                                            ; Return: ES:BX -> DOS list of lists
                                    ;mov    ax,[es:8Ch]
0000354A 26A18C00                   mov     ax,[es:UMB_HeadIdx]     ; And read what's in ES:008C
0000354E 83F8FF                     cmp     ax,0FFFFh
                                    ;je     short xhu_e             ; If it's 0xFFFF, it's an error...
                                    ;clc                            ; Else, it isn't.
                                    ;jmp    short xhu_x
                                xhu_e:
                                    ;stc
00003551 F5                         cmc     ; 09/04/2019 - Retro DOS v4.0 ; *
                                xhu_x:
                                    ;pop    bx ; *
                                    ;pop    es
                                    ;pop    ds ; ?
                                    ;pop    si ; ?
00003552 C3                         retn

                                ; ----------------------------------------------------------------------------
                                ;*** linkumb - links UMBs not already linked in; updates fm_umb as needed
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:    None
                                ; EXIT:     fm_umb == 0 if not linked in previously, 1 if already linked in
                                ; ERROR:    None
                                ; USES:     AX, BX, fm_umb
                                ; ----------------------------------------------------------------------------

                                linkumb:
00003553 B80258                     mov     ax,DOS_GET_UMBLINK ; 5802h
00003556 CD21                       int     21h                     ; Current link-state is now in al

                                    or      al,al                   ; BUGBUG: proper check?
0000355A 7508                       jnz     short lumbX             ; Jumps if UMBs already linked in

0000355C B80358                     mov     ax,DOS_SET_UMBLINK ; 5803h
0000355F BB0100                     mov     bx,1
00003562 CD21                       int     21h
                                lumbX:
00003564 C3                         retn

                                ;%endif

                                ; 01/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ; (SYSINIT:2B5Fh)

                                ; ----------------------------------------------------------------------
                                ; SYSCONF.ASM (MSDOS 6.0 - 1991)
                                ; ----------------------------------------------------------------------
                                ; 09/04/2019 - Retro DOS v4.0

                                ;----------------------------------------------------------------------------
                                ;
                                ; procedure : InitDevLoad
                                ;
                                ;   Input : DeviceHi = 0 indicates load DD in low memory
                                ;                    = 1 indicates load in UMB:
                                ;                       ConvLoad = 0 indicates a new-style load (see below)
                                ;                                = 1 indicates a DOS 5-style load
                                ;           DevSize  = Size of the device driver file in paras
                                ;
                                ;   Output : none
                                ;
                                ;   Initializes DevLoadAddr, DevLoadEnd & DevEntry.
                                ;   Also sets up a header for the Device driver entry for mem utility
                                ;
                                ;----------------------------------------------------------------------------
                                ; For a "new-style load", we break off the current DevEntry and link the umbs
                                ; as we see fit, using HideUMBs (and UnHideUMBs at exit, though _it_ decides
                                ; whether it's entitled to do anything). HideUMBs uses the chart built by
                                ; ParseVar to determine which UMBs to leave FREE, and which not.
                                ;----------------------------------------------------------------------------

                                    ; 01/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:364Ah)
                                InitDevLoad:
                                    ; 01/01/2023
                                    ;push   es ; *
                                    ; 11/12/2022
                                    ; ds = cs
00003565 803E[5124]00               cmp     byte [DeviceHi],0
                                    ;cmp    byte [cs:DeviceHi],0  ; Are we loading in UMB ?
                                    ;je     short InitForLo             ; no, init for lo mem
0000356A 7439                       je      short initforlo_x ; 09/04/2019

                                ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ; %if 0
                                    ; 01/01/2023
0000356C 803E[4124]01               cmp     byte [ConvLoad],1     ; Are we loading as per DOS 5?
                                    ;cmp    byte [cs:ConvLoad],1  ; Are we loading as per DOS 5?
00003571 7413                       je      short InitForConv

                                ; There are two stages to preparing upper-memory; first, we mark as 8+HIDDEN
                                ; any areas not specified on the /L:... chain. Second, we mark as 8+FROZEN
                                ; any areas left in upper-memory, except for elements in the load UMB...
                                ; we then malloc space as per Dos-5 style, and mark as free any spaces which
                                ; are 8+FROZEN (but leave 8+HIDDEN still hidden). The load is performed,
                                ; and UnHideUMBs later on marks all 8+HIDDEN as free.

00003573 E85904                     call    ShrinkUMB               ; Stop using the old device arena

00003576 E892FC                     call    HideUMBs                ; Mark up the UM area as we see fit
00003579 E8CBFE                     call    FreezeUM                ; Hide everything BUT the load area
0000357C E85700                     call    GetUMBForDev            ; And grab that load area as needed
0000357F 9C                         pushf
00003580 E851FE                     call    UnFreeze                ; Then unhide everything frozen
00003583 9D                         popf
                                    ;jc     short InitForLo                 ; (if carry, it's loading low)
                                    ;jmp    short InitForHi
                                    ; 06/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
00003584 EB0B                       jmp     short idl0

                                ;%endif ; 01/11/2022
```

```
                                    ; 01/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:2B67h)
                                InitForConv:
                                    ; 11/12/2022
                                    ; ds = cs
00003586 E83700                     call    SpaceInUMB              ; Do we have space left in the
                                                                    ;  current UMB ?
00003589 7308                       jnc     short InitForHi             ; yes, we have
0000358B E84104                     call    ShrinkUMB               ; shrink the current UMB in use
0000358E E84500                     call    GetUMBForDev            ; else try to allocate new UMB
                                idl0: ; 06/07/2023
00003591 720D                       jc      short InitForLo             ; we didn't succeed, so load
                                                                    ;  in low memory
                                InitForHi:
                                    ; 11/12/2022
                                    ; ds = cs
                                    ;mov    ax,[cs:DevUMBFree]      ; get Para addr of free mem
                                    ;mov    dx,[cs:DevUMBAddr]      ; UMB start addr
                                    ;add    dx,[cs:DevUMBSize]      ; DX = UMB End addr
00003593 A1[4724]                   mov     ax,[DevUMBFree]
00003596 8B16[4324]                 mov     dx,[DevUMBAddr]
0000359A 0316[4524]                 add     dx,[DevUMBSize]
0000359E EB0C                       jmp     short idl1

                                InitForLo:
                                    ; 11/12/2022
                                    ; ds = cs
                                    ;mov    byte [cs:DeviceHi],0    ; in case we failed to load
000035A0 C606[5124]00               mov     byte [DeviceHi],0
                                initforlo_x:
                                    ; 11/12/2022
                                    ; ds = cs
                                                                    ;  into UMB indicate that
                                                                    ;  we are loading low
                                    ;mov    ax,[cs:memhi]           ; AX = start of Low memory
                                    ;mov    dx,[cs:ALLOCLIM]        ; DX = End of Low memory
000035A5 A1[6403]                   mov     ax,[memhi]
000035A8 8B16[A502]                 mov     dx,[ALLOCLIM]
                                idl1:
000035AC E86600                     call    DevSetMark              ; setup a sub-arena for DD
                                    ; 11/12/2022
                                    ; ds = cs
                                    ;mov    [cs:DevLoadAddr],ax     ; init the Device load address
                                    ;mov    [cs:DevLoadEnd],dx      ; init the limit of the block
                                    ;mov    word [cs:DevEntry],0    ; init Entry point to DD
000035AF A3[3524]                   mov     [DevLoadAddr],ax
000035B2 8916[3724]                 mov     [DevLoadEnd],dx
000035B6 C706[3924]0000             mov     word [DevEntry],0
000035BC A3[3B24]                   mov     [DevEntry+2],ax
                                    ; 01/01/2023
                                    ;pop    es ; *
000035BF C3                         retn

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : SpaceInUMB?
                                ;
                                ;   Input : DevUMBAddr, DevUMBSize, DevUMBFree & DevSize
                                ;   Output : Carry set if no space in UMB
                                ;            Carry clear if Space is available for the device in
                                ;              current UMB
                                ;
                                ;--------------------------------------------------------------------------

                                SpaceInUMB:
                                    ; 11/12/2022
                                    ; ds = cs
                                    ;mov    ax,[cs:DevUMBSize]
                                    ;add    ax,[cs:DevUMBAddr]      ; End of UMB
                                    ;sub    ax,[cs:DevUMBFree]      ; - Free = Remaining space
000035C0 A1[4524]                   mov     ax,[DevUMBSize]
000035C3 0306[4324]                 add     ax,[DevUMBAddr]             ; End of UMB
000035C7 2B06[4724]                 sub     ax,[DevUMBFree]             ; - Free = Remaining space
                                    ; 11/12/2022
                                    ;or     ax,ax                   ; Nospace ?
                                    ;jnz    short spcinumb1
                                    ;stc
                                    ;retn
                                    ; 11/12/2022
000035CB 83F801                     cmp     ax,1
000035CE 7205                       jb      short spcinumb2         ; cf=1
                                spcinumb1:
000035D0 48                         dec     ax                      ; space for sub-arena
                                    ; 11/12/2022
                                    ; ds = cs
000035D1 3B06[3324]                 cmp     ax,[DevSize]
                                    ;cmp    ax,[cs:DevSize]             ; do we have space ?
                                spcinumb2:
000035D5 C3                         retn

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : PrepareMark
                                ;
                                ;   Input : AX==Address of MCB (not addr of free space), BX==Size
                                ;   Output : None; MCB marked appropriately and DevUMB* set as needed.
                                ;
                                ;--------------------------------------------------------------------------

                                ;   ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;
                                ;PrepareMark:
                                ;   push    ds
                                ;   mov     ds,ax
                                ;   mov     word [ARENA.OWNER],8
                                ;   mov     word [ARENA.NAME],'SD' ; 4453h
                                ;   pop     ds
                                ;
                                ;   inc     ax
                                ;   mov     [cs:DevUMBAddr],ax
                                ;   mov     [cs:DevUMBFree],ax
                                ;   mov     [cs:DevUMBSize],bx      ; update the UMB Variables
                                ;   retn

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : GetUMBForDev
                                ;
                                ;   Input : DevSize
                                ;   Output : Carry set if couldn't allocate a UMB to fit the
                                ;            the device.
                                ;            If success carry clear
                                ;
```

```
                                ;   Allocates the biggest UMB for loading devices and updates
                                ;   DevUMBSize, DevUMBAddr & DevUMBFree if it succeeded in allocating
                                ;   UMB.
                                ;
                                ;   This routine relies on the fact that all of the low memory
                                ;   is allocated, and any DOS alloc calls should return memory
                                ;   from the UMB pool.
                                ;
                                ;--------------------------------------------------------------------------
                                    ; 01/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:2BC6h)

                                GetUMBForDev:
                                    ; 11/12/2022
                                    ; ds = cs
000035D6 BBFFFF                     mov     bx,0FFFFh
000035D9 B80048                     mov     ax,4800h
000035DC CD21                       int     21h
                                            ; DOS - 2+ - ALLOCATE MEMORY
                                            ; BX = number of 16-byte paragraphs desired

000035DE 09DB                       or      bx,bx
                                    ;jz     short gufd_err
                                    ; 09/09/2023
000035E0 742E                       jz      short gufd_error ; bx = 0

000035E2 4B                         dec     bx
                                    ; 11/12/2022
                                    ; ds = cs
000035E3 391E[3324]                 cmp     [DevSize],bx
                                    ;cmp    [cs:DevSize],bx
000035E7 7725                       ja      short gufd_err

000035E9 43                         inc     bx

000035EA B80048                     mov     ax,4800h
000035ED CD21                       int     21h
000035EF 721D                       jc      short gufd_err

                                    ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;dec    ax
                                    ;call   PrepareMark
                                    ;
                                PrepareMark:
000035F1 1E                         push    ds
000035F2 48                         dec     ax
000035F3 8ED8                       mov     ds,ax
000035F5 C70601000800               mov     word [ARENA.OWNER],8
000035FB C70608005344               mov     word [ARENA.NAME],'SD' ; 4453h
00003601 40                         inc     ax
00003602 1F                         pop     ds
                                    ; 11/12/2022
                                    ; ds = cs
                                    ;mov    [cs:DevUMBSize],bx    ; update the UMB Variables
                                    ;mov    [cs:DevUMBAddr],ax
                                    ;mov    [cs:DevUMBFree],ax
                                gufd_x:             ; 09/09/2023
00003603 891E[4524]                 mov     [DevUMBSize],bx               ; update the UMB Variables
00003607 A3[4324]                   mov     [DevUMBAddr],ax
0000360A A3[4724]                   mov     [DevUMBFree],ax
                                    ;
                                    ; 11/12/2022
                                    ; cf=0
                                    ;clc                           ; mark no error
0000360D C3                         retn

                                    ; 09/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
                                %if 1
                                gufd_err:
0000360E 31DB                       xor     bx,bx ; 0
                                gufd_error:
00003610 31C0                       xor     ax,ax ; 0
00003612 F9                         stc     ; cf=1
00003613 EBEE                       jmp     short gufd_x
                                %endif

                                ; 09/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
                                %if 0
                                gufd_err:
                                    xor     ax,ax ; 0
                                    ; 11/12/2022
                                    ; ds = cs
                                    ;mov    [cs:DevUMBSize],ax    ; erase the previous values
                                    ;mov    [cs:DevUMBAddr],ax
                                    ;mov    [cs:DevUMBFree],ax
                                    mov     [DevUMBSize],ax               ; erase the previous values
                                    mov     [DevUMBAddr],ax
                                    mov     [DevUMBFree],ax
                                    stc
                                    retn
                                %endif

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : DevSetMark
                                ;
                                ;   Input : AX - Free segment were device is going to be loaded
                                ;   Output : AX - Segment at which device can be loaded (AX=AX+1)
                                ;
                                ;   Creates a sub-arena for the device driver
                                ;   puts 'D' marker in the sub-arena
                                ;   Put the owner of the sub-arena as (AX+1)
                                ;   Copies the file name into sub-arena name field
                                ;
                                ;   Size field of the sub-arena will be set only at succesful
                                ;   completion of Device load.
                                ;
                                ;--------------------------------------------------------------------------

                                    ; 01/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:2C13h)

                                DevSetMark:
00003615 06                         push    es
                                    ; 03/01/2023
                                    ;push   di
00003616 1E                         push    ds
00003617 56                         push    si
00003618 8EC0                       mov     es,ax
0000361A 26C606000044               mov     byte [es:devmark.id],devmark_device ; 'D'
00003620 40                         inc     ax
00003621 26A30100                   mov     [es:devmark.seg],ax

                                ;-------------- Copy file name
```

```
40141
40142 00003625 50                      push    ax                      ; save load addr
40143
40144                                  ; 09/09/2023
40145                                  ; ds = cs
40146                                  ;lds    si,[cs:bpb_addr]        ; command line is still there
40147 00003626 C536[8103]              lds     si,[bpb_addr]
40148
40149 0000362A 89F7                    mov     di,si
40150 0000362C FC                      cld
40151                                dsm_again:
40152 0000362D AC                      lodsb
40153 0000362E 3C3A                    cmp     al,':'
40154 00003630 7504                    jne     short isit_slash
40155 00003632 89F7                    mov     di,si
40156 00003634 EBF7                    jmp     short dsm_again
40157                                isit_slash:
40158 00003636 3C5C                    cmp     al, '\'
40159 00003638 7504                    jne     short isit_null
40160 0000363A 89F7                    mov     di,si
40161 0000363C EBEF                    jmp     short dsm_again
40162                                isit_null:
40163 0000363E 08C0                    or      al,al
40164 00003640 75EB                    jnz     short dsm_again
40165 00003642 89FE                    mov     si,di
40166
40167 00003644 BF0800                  mov     di,devmark.filename ; 8
40168 00003647 B90800                  mov     cx,8                    ; maximum 8 characters
40169                                dsm_next_char:
40170 0000364A AC                      lodsb
40171 0000364B 08C0                    or      al, al
40172 0000364D 7407                    jz      short blankout
40173 0000364F 3C2E                    cmp     al, '.'
40174 00003651 7403                    je      short blankout
40175 00003653 AA                      stosb
40176 00003654 E2F4                    loop    dsm_next_char
40177                                blankout:
40178 00003656 E304                    jcxz    dsm_exit
40179 00003658 B020                    mov     al, ' '
40180 0000365A F3AA                    rep     stosb                   ; blank out the rest
40181                                dsm_exit:
40182 0000365C 58                      pop     ax                      ; restore load addr
40183 0000365D 5E                      pop     si
40184 0000365E 1F                      pop     ds
40185                                  ; 03/01/2023
40186                                  ;pop    di
40187 0000365F 07                      pop     es
40188 00003660 C3                      retn
40189
40190                                ;--------------------------------------------------------------------------
40191                                ;
40192                                ; procedure : SizeDevice
40193                                ;
40194                                ;   Input : ES:SI - points to device file to be sized
40195                                ;
40196                                ;   Output : Carry set if file cannot be opened or if it is an OS2EXE file
40197                                ;
40198                                ;   Calculates the size of the device file in paras and stores it
40199                                ;   in DevSize
40200                                ;
40201                                ;--------------------------------------------------------------------------
40202
40203                                  ; 01/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
40204                                SizeDevice:
40205                                  ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
40206                                  ; 11/12/2022 ; *
40207 00003661 1E                      push    ds ; *
40208 00003662 06                      push    es
40209 00003663 1F                      pop     ds
40210 00003664 89F2                    mov     dx,si                   ; ds:dx -> file name
40211 00003666 B8003D                  mov     ax,3D00h                ; open
40212 00003669 CD21                    int     21h
40213 0000366B 7237                    jc      short sd_err            ; open failed
40214
40215 0000366D 89C3                    mov     bx,ax                   ; BX - file handle
40216 0000366F B80242                  mov     ax,4202h                ; seek
40217 00003672 31C9                    xor     cx,cx
40218 00003674 89CA                    mov     dx,cx                   ; to end of file
40219 00003676 CD21                    int     21h
40220 00003678 7223                    jc      short sd_close          ; did seek fail (impossible)
40221 0000367A 83C00F                  add     ax,15                   ; para convert
40222 0000367D 83D200                  adc     dx,0
40223 00003680 F7C2F0FF                test    dx,0FFF0h               ; size > 0ffffh paras ?
40224                                  ;jz     short szdev1            ; no
40225                                  ; 22/07/2023
40226 00003684 7409                    jz      short sd_ctp
40227 00003686 2EC706[3324]FFFF        mov     word [cs:DevSize],0FFFFh ; invalid device size
40228                                                                  ; assuming that we fail later
40229 0000368D EB0E                    jmp     short sd_close
40230                                sd_ctp:
40231                                  ; 22/07/2023
40232                                ;szdev1:
40233 0000368F B104                    mov     cl,4                    ; convert it to paras
40234 00003691 D3E8                    shr     ax,cl
40235 00003693 B10C                    mov     cl,12
40236 00003695 D3E2                    shl     dx,cl
40237 00003697 09D0                    or      ax,dx ; * ; cf=0
40238                                  ;
40239                                  ; 22/07/2023 - Retro DOS v4.2 IO.SYS (optimized)
40240                                  ; MSDOS 6.21 IO.SYS - SYSINIT:37A6h
40241                                  ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
40242                                  ;cmp    ax,[cs:DevSizeOption]
40243                                  ;ja     short szdev2
40244                                  ;mov    ax,[cs:DevSizeOption]
40245                                  ; 12/12/2022
40246                                  ;clc
40247                                ;szdev2:
40248 00003699 2EA3[3324]              mov     [cs:DevSize],ax             ; save file size (in paragraps)
40249                                  ; 22/07/2023
40250                                  ;clc ; cf=0 ; *       ; CLC is not needed here
40251                                                  ; (OR instruction clears CF) - E.TAN 22/07/2023
40252
40253                                  ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
40254                                  ; 12/12/2022
40255                                  ; cf=0
40256                                  ;clc
40257                                sd_close:
40258 0000369D 9C                      pushf                           ; let close not spoil our
40259                                                                  ;  carry flag
40260 0000369E B8003E                  mov     ax,3E00h                ; close
40261 000036A1 CD21                    int     21h                     ; we are not checking for err
40262 000036A3 9D                      popf
40263                                sd_err:
40264                                  ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
```

```
                                    ; 11/12/2022 ; *
000036A4 1F                         pop     ds ; *
000036A5 C3                         retn

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : ExecDev
                                ;
                                ;   Input : ds:dx -> device to be executed
                                ;           DevLoadAddr - contains where device has to be loaded
                                ;
                                ;   Output : Carry if error
                                ;            Carry clear if no error
                                ;
                                ;   Loads a device driver using the 4b03h function call
                                ;
                                ;--------------------------------------------------------------------------

                                    ; 01/11/2022
                                ExecDev:
000036A6 2E8B1E[3524]               mov     bx,[cs:DevLoadAddr]
000036AB 2E891E[4D24]               mov     [cs:DevExecAddr],bx    ; Load the parameter block
                                                                   ;  block for exec with
                                                                   ;  load address
000036B0 2E891E[4F24]               mov     [cs:DevExecReloc],bx
000036B5 8CCB                       mov     bx,cs
000036B7 8EC3                       mov     es,bx
000036B9 BB[4D24]                   mov     bx,DevExecAddr         ; es:bx points to parameters
                                    ;mov    al,3    ; (load program only)
                                    ;mov    ah,EXEC ; 4Bh
                                    ; 04/07/2023
000036BC B8034B                     mov     ax,(EXEC<<8)|03h
000036BF CD21                       int     21h                    ; load in the device driver
                                            ; DOS - 2+ - LOAD OR EXECUTE (EXEC)
                                            ; DS:DX -> ASCIZ filename
                                            ; ES:BX -> parameter block
                                            ; AL = subfunction
000036C1 C3                         retn

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : RetFromUM
                                ;
                                ;   Input : None
                                ;   Output : ConvLoad set if didn't previously call HideUMBs
                                ;            ConvLoad clear if did.
                                ;
                                ;   Prepares memory for more devices after returning from loading one
                                ;   using the DOS 6 options (/L:... etc).
                                ;
                                ;--------------------------------------------------------------------------

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ;  (SYSINIT:37D1h)

                                ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                RetFromUM:
                                    ; 31/12/2022
                                    ; ds = cs
000036C2 9C                         pushf
                                    ;mov    byte [cs:ConvLoad],1
000036C3 C606[4124]01               mov     byte [ConvLoad],1
000036C8 E8ECFD                     call    UnHideUMBs
000036CB 7204                       jc      short rfUM1            ; Skip this if didn't HideUMBs
                                    ; 31/12/2022
                                    ; ds = cs
                                    ;;mov   byte [cs:ConvLoad],0
                                    ;mov    byte [ConvLoad],0
                                    ; 09/09/2023
000036CD FE0E[4124]                 dec     byte [ConvLoad] ; -> 0
                                rfUM1:
000036D1 9D                         popf
000036D2 C3                         retn

                                ;%endif ; 01/11/2022

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : RemoveNull
                                ;
                                ;   Input : ES:SI points to a null terminated string
                                ;
                                ;   Output : none
                                ;
                                ;   Replaces the null at the end of a string with blank
                                ;
                                ;--------------------------------------------------------------------------

                                    ; 01/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:2CCEh)
                                RemoveNull:
                                    ; 11/12/2022
                                    ; ds = cs
                                rn_next:
000036D3 268A1C                     mov     bl,[es:si]
000036D6 08DB                       or      bl,bl                  ; null ?
000036D8 7403                       jz      short rn_gotnull
000036DA 46                         inc     si                     ; advance the pointer
000036DB EBF6                       jmp     short rn_next
                                rn_gotnull:
                                    ; 11/12/2022
000036DD 8A1E[6624]                 mov     bl,[DevSavedDelim]
                                    ;mov    bl,[cs:DevSavedDelim]
000036E1 26881C                     mov     [es:si],bl             ; replace null with blank
                                    ; 02/11/2022
                                ; 11/12/2022
                                rba_ok:             ; 10/04/2019
000036E4 C3                         retn

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : RoundBreakAddr
                                ;
                                ;   Input : DevBrkAddr
                                ;   Output : DevBrkAddr
                                ;
                                ;   Rounds DevBrkAddr to a para address so that it is of the form xxxx:0
                                ;
                                ;--------------------------------------------------------------------------

                                RoundBreakAddr:
000036E5 2EA1[3D24]                 mov     ax,[cs:DevBrkAddr]
000036E9 E829DC                     call    ParaRound
```

```
40389 000036EC 2E0106[3F24]               add     [cs:DevBrkAddr+2],ax
40390 000036F1 2EC706[3D24]0000           mov     word [cs:DevBrkAddr],0
40391 000036F8 2EA1[3724]                 mov     ax,[cs:DevLoadEnd]
40392 000036FC 2E3906[3F24]               cmp     [cs:DevBrkAddr+2],ax
40393 00003701 76E1                       jbe     short rba_ok
40394 00003703 E92911                     jmp     mem_err
40395                                     ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
40396                                     ; 11/12/2022
40397                                   ;rba_ok:
40398                                   ;   retn
40399
40400                                   ;--------------------------------------------------------------------------
40401                                   ;
40402                                   ; procedure : DevSetBreak
40403                                   ;
40404                                   ;   Input : DevBrkAddr
40405                                   ;   Output : Carry set if Device returned Init failed
40406                                   ;            Else carry clear
40407                                   ;
40408                                   ;--------------------------------------------------------------------------
40409
40410                                   DevSetBreak:
40411 00003706 50                         push    ax
40412
40413 00003707 2EA1[3F24]                  mov     ax,[cs:DevBrkAddr+2]   ;remove the init code
40414 0000370B 2E803E[6619]00             cmp     byte [cs:multdeviceflag],0
40415 00003711 750F                       jne     short set_break_continue ;do not check it.
40416 00003713 2E3B06[3524]               cmp     ax,[cs:DevLoadAddr]
40417 00003718 7508                       jne     short set_break_continue ;if not same, then o.k.
40418
40419                                     ;cmp    word [cs:DevBrkAddr],0
40420                                     ;je     short break_failed    ;[DevBrkAddr+2]=[memhi] & [DevBrkAddr]=0
40421                                     ; 12/12/2022
40422 0000371A 2E833E[3D24]01             cmp     word [cs:DevBrkAddr],1
40423 00003720 7204                       jb      short break_failed
40424
40425                                   set_break_continue:
40426 00003722 E8C0FF                     call    RoundBreakAddr
40427                                     ; 12/12/2022
40428 00003725 F8                         clc
40429                                   break_failed:
40430 00003726 58                         pop     ax
40431                                     ;clc
40432 00003727 C3                         retn
40433
40434                                     ; 12/12/2022
40435                                   ;break_failed:
40436                                     ;pop    ax
40437                                     ;stc
40438                                     ;retn
40439
40440                                   ;--------------------------------------------------------------------------
40441                                   ;
40442                                   ; procedure : DevBreak
40443                                   ;
40444                                   ;   Input : DevLoadAddr & DevBrkAddr
40445                                   ;   Output : none
40446                                   ;
40447                                   ;   Marks a succesful install of a device driver
40448                                   ;   Sets device size field in sub-arena &
40449                                   ;   Updates Free ptr in UMB or adjusts memhi
40450                                   ;
40451                                   ;--------------------------------------------------------------------------
40452
40453                                     ; 11/12/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
40454                                   DevBreak:
40455                                     ;push  ds ; 11/12/2022
40456
40457                                     ; 11/12/2022
40458 00003728 0E                         push    cs
40459 00003729 1F                         pop     ds
40460                                     ;mov    ax,[cs:DevLoadAddr]
40461                                     ;mov    bx,[cs:DevBrkAddr+2]
40462 0000372A A1[3524]                   mov     ax,[DevLoadAddr]
40463 0000372D 8B1E[3F24]                 mov     bx,[DevBrkAddr+2]
40464                                     ; 11/12/2022
40465 00003731 1E                         push    ds
40466
40467 00003732 48                         dec     ax                      ; seg of sub-arena
40468 00003733 8ED8                       mov     ds,ax
40469 00003735 40                         inc     ax                      ; Back to Device segment
40470 00003736 29D8                       sub     ax,bx
40471 00003738 F7D8                       neg     ax                      ; size of device in paras
40472 0000373A A30300                     mov     [devmark.size],ax       ; store it in sub-arena
40473
40474                                     ; 11/12/2022
40475 0000373D 1F                         pop     ds
40476                                     ; ds = cs
40477
40478 0000373E 803E[5124]00               cmp     byte [DeviceHi],0
40479                                     ;cmp    byte [cs:DeviceHi],0
40480 00003743 7405                       je      short db_lo
40481                                     ;mov    [cs:DevUMBFree],bx    ; update Free ptr in UMB
40482                                     ;jmp    short db_exit
40483                                     ; 11/12/2022
40484 00003745 891E[4724]                 mov     [DevUMBFree],bx
40485 00003749 C3                         retn
40486                                   db_lo:
40487                                     ; 11/12/2022
40488                                     ; ds = cs
40489                                     ;mov    [cs:memhi],bx
40490                                     ;mov    word [cs:memlo],0
40491 0000374A 891E[6403]                 mov     [memhi],bx
40492 0000374E C706[6203]0000             mov     word [memlo],0 ; 18/12/2022
40493                                   db_exit:
40494                                     ;pop    ds ; 11/12/2022
40495                                   sd_ret:             ; 09/09/2023
40496 00003754 C3                         retn
40497
40498                                   ; 10/04/2019 - Retro DOS v4.0
40499
40500                                   ;--------------------------------------------------------------------------
40501                                   ;
40502                                   ; procedure : ParseSize
40503                                   ;
40504                                   ;   Parses the command line for SIZE= command
40505                                   ;
40506                                   ;   ES:SI = command line to parsed
40507                                   ;
40508                                   ;   returns ptr to command line after SIZE= option in ES:SI
40509                                   ;   updates the DevSizeOption variable with value supplied
40510                                   ;   in SIZE=option
40511                                   ;   Returns carry if the SIZE option was invalid
40512                                   ;
```

```
40513                                   ;--------------------------------------------------------------------------
40514
40515                                       ; 02/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
40516                                       ; (SYSINIT:2D5Ah)
40517
40518                                       ; 09/09/2023 - Retro DOS v4.2 IO.SYS (Optimization) ((&BugFix))
40519                                       ; (MSDOS 6.21 IO.SYS - SYSINIT:3871h) - Retro DOS v4.2 -
40520                                       ; (PCDOS 7.1 IO.SYS - SYSINIT:3D6Eh)  - Retro DOS v5.0 -
40521                                   ParseSize:
40522                                       ;push  bx
40523                                       ;mov   bx,si
40524
40525                                       ; 09/09/2023
40526 00003755 56                         push   si ; * ; mov bx,si
40527
40528                                       ; 11/12/2022
40529                                       ; ds = cs
40530                                       ;mov   word [cs:DevSizeOption],0 ; init the value
40531                                       ;mov   [cs:DevCmdLine],si
40532                                       ;mov   [cs:DevCmdLine+2],es
40533 00003756 C706[5224]0000             mov    word [DevSizeOption],0 ; init the value
40534 0000375C 8936[6224]                 mov    [DevCmdLine],si
40535 00003760 8C06[6424]                 mov    [DevCmdLine+2],es
40536 00003764 E82400                     call   SkipDelim
40537 00003767 26813C5349                 cmp    word [es:si],'SI' ; 4953h
40538 0000376C 7528                       jne    short ps_no_size
40539 0000376E 26817C025A45               cmp    word [es:si+2],'ZE' ; 455Ah
40540 00003774 7520                       jne    short ps_no_size
40541 00003776 268A4404                   mov    al,[es:si+4]
40542 0000377A E80D10                     call   delim
40543                                     ;jne   short ps_no_size
40544                                     ; 22/07/2023
40545 0000377D 7518                       jne    short ps_no_size_2 ; cf=0 here
40546 0000377F 83C605                     add    si,5
40547 00003782 E81400                     call   GetHexNum
40548 00003785 7210                       jc     short ps_err
40549                                     ; 11/12/2022
40550                                     ; ds = cs
40551                                     ;mov   [cs:DevSizeOption],ax
40552 00003787 A3[5224]                   mov    [DevSizeOption],ax
40553
40554                                     ; 09/09/2023
40555 0000378A 58                         pop    ax  ; * (discard previous si value on top of stack)
40556
40557                                   ;   call   SkipDelim ; **
40558                                   ;
40559                                   ;   ; 22/07/2023
40560                                   ;;ps_no_size_2:
40561                                   ;   ; cf = 0
40562                                   ;   retn
40563
40564                                       ; 09/09/2023
40565                                       ;jmp   short SkipDelim
40566
40567                                   ; 09/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
40568                                   %if 1
40569                                       ; 01/11/2022
40570                                   SkipDelim:
40571                                   sd_next_char:
40572 0000378B 268A04                     mov    al,[es:si]
40573 0000378E E8F90F                     call   delim
40574 00003791 75C1                       jnz    short sd_ret ; cf=0 ; 09/09/2023
40575 00003793 46                         inc    si
40576 00003794 EBF5                       jmp    short sd_next_char ; 01/11/2022
40577                                     ; 11/12/2022
40578                                     ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
40579                                   ;sd_ret:
40580                                     ;retn
40581                                   %endif
40582
40583                                     ;;;call SkipDelim ; **
40584                                     ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
40585                                     ;mov   bx,si
40586                                   ps_no_size:
40587                                     ;mov   si,bx
40588                                     ;pop   bx
40589 00003796 F8                       clc    ; cf=0
40590                                   ;retn
40591                                   ; 11/12/2022
40592                                   ps_err:             ; cf=1
40593                                   ps_no_size_2:       ; 09/09/2023 (cf=0)
40594                                       ; 09/09/2023
40595 00003797 5E                         pop    si ; * ; mov si,bx
40596                                   ;sd_ret:    ; cf=?
40597 00003798 C3                         retn
40598
40599                                   ;ps_err:
40600                                       ; 02/11/2022
40601                                       ;pop   bx
40602                                       ;stc
40603                                       ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
40604                                       ; 11/12/2022
40605                                       ; cf=1
40606                                       ;stc
40607                                       ; 11/12/2022
40608                                   ;sd_ret:
40609                                       ; 22/07/2023
40610                                       ; 12/04/2019
40611                                       ;retn
40612
40613                                   ; 12/04/2019 - Retro DOS v4.0
40614
40615                                   ;--------------------------------------------------------------------------
40616                                   ;
40617                                   ; procedure : SkipDelim
40618                                   ;
40619                                   ;   Skips delimiters in the string pointed to by ES:SI
40620                                   ;   Returns ptr to first non-delimiter character in ES:SI
40621                                   ;
40622                                   ;--------------------------------------------------------------------------
40623
40624                                   ; 09/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
40625                                   %if 0
40626                                       ; 01/11/2022
40627                                   SkipDelim:
40628                                   sd_next_char:
40629                                       mov    al,[es:si]
40630                                       call   delim
40631                                       jnz    short sd_ret
40632                                       inc    si
40633                                       jmp    short sd_next_char ; 01/11/2022
40634                                       ; 11/12/2022
40635                                       ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
40636                                   ;sd_ret:
```

```
                                    ;retn
                                %endif

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : GetHexNum
                                ;
                                ;   Converts an ascii string terminated by a delimiter into binary.
                                ;   Assumes that the ES:SI points to a Hexadecimal string
                                ;
                                ;   Returns in AX the number number of paras equivalent to the
                                ;   hex number of bytes specified by the hexadecimal string.
                                ;
                                ;   Returns carry in case it encountered a non-hex character or
                                ;   if it encountered crlf
                                ;
                                ;--------------------------------------------------------------------------

                                ; 13/05/2019

                                    ; 03/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:38C5h)

                                    ; 01/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:2DA5h)
                                GetHexNum:
00003799 31C0                       xor     ax,ax
0000379B 31D2                       xor     dx,dx
                                ghn_next:
0000379D 268A1C                     mov     bl,[es:si]
000037A0 80FB0D                     cmp     bl,cr  ; 0Dh
000037A3 7436                       je      short ghn_err
000037A5 80FB0A                     cmp     bl,lf  ; 0Ah
000037A8 7431                       je      short ghn_err
000037AA 50                         push    ax
000037AB 88D8                       mov     al,bl
000037AD E8DA0F                     call    delim
000037B0 58                         pop     ax
                                    ; 03/01/2023
000037B1 B90400                     mov     cx,4
000037B4 7410                       jz      short ghn_into_paras
000037B6 E82400                     call    GetNibble
                                    ;jc     short ghn_err
                                    ; 11/12/2022
000037B9 7221                       jc      short ghn_ret ; cf=1
                                    ; 03/01/2023
                                    ;mov    cx,4
                                ghn_shift1:
000037BB D1E0                       shl     ax,1
000037BD D1D2                       rcl     dx,1
000037BF E2FA                       loop    ghn_shift1
000037C1 08D8                       or      al,bl
000037C3 46                         inc     si
000037C4 EBD7                       jmp     short ghn_next
                                ghn_into_paras:
000037C6 83C00F                     add     ax,15
000037C9 83D200                     adc     dx,0
000037CC F7C2F0FF                   test    dx,0FFF0h
000037D0 7509                       jnz     short ghn_err
                                    ; 03/01/2023
                                    ;mov    cx,4
                                ghn_shift2:
000037D2 F8                         clc
000037D3 D1DA                       rcr     dx,1
000037D5 D1D8                       rcr     ax,1
000037D7 E2F9                       loop    ghn_shift2
000037D9 F8                         clc
000037DA C3                         retn
                                    ; 11/12/2022
                                ghn_err:
                                gnib_err:
000037DB F9                         stc
                                ghn_ret:
                                gnib_ret:
000037DC C3                         retn

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : GetNibble
                                ;
                                ;   Convert one nibble (hex digit) in BL into binary
                                ;
                                ;   Returns binary value in BL
                                ;
                                ;   Returns carry if BL contains non-hex digit
                                ;
                                ;--------------------------------------------------------------------------

                                GetNibble:
000037DD 80FB30                     cmp     bl,'0'
                                    ;jb     short gnib_err
                                    ; 11/12/2022
000037E0 72FA                       jb      short gnib_ret ; cf=1
000037E2 80FB39                     cmp     bl,'9'
000037E5 7704                       ja      short is_it_hex
000037E7 80EB30                     sub     bl,'0'          ; clc
000037EA C3                         retn
                                is_it_hex:
000037EB 80FB41                     cmp     bl,'A'
                                    ;jb     short gnib_err
                                    ; 11/12/2022
000037EE 72EC                       jb      short gnib_ret ; cf=1
000037F0 80FB46                     cmp     bl,'F'
000037F3 77E6                       ja      short gnib_err ; 11/12/2022
000037F5 80EB37                     sub     bl,'A'- 10     ; clc
000037F8 C3                         retn

                                    ; 11/12/2022
                                ;gnib_err:
                                ;   stc
                                ;gnib_ret:
                                ;   retn

                                ;============================================================================

                                ; 12/04/2019 - Retro DOS v4.0

                                ; umb.inc (MSDOS 6.0, 1991)
                                DOS_ARENA   equ 24h         ; offset of arena_head var in DOS data segm.
                                UMB_ARENA   equ 8Ch         ; offset of umb_head in DOS data

                                XMM_REQUEST_UMB    equ 10h
                                XMM_RELEASE_UMB    equ 11h
```

```
                                ; 09/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)

                                ;--------------------------------------------------------------------------
                                ;
                                ; Procedure Name   : umb_insert
                                ;
                                ; Inputs           : DOSDATA:UMB_HEAD = start of umb chain
                                ;                  : BX = seg address of UMB to be linked in
                                ;                  : DX = size of UMB to be linked in paras
                                ;                  ; DS = data
                                ;
                                ; Outputs          : links the UMB into the arena chain
                                ;
                                ; Uses                    : AX, CX, ES, DX, BX
                                ;
                                ;--------------------------------------------------------------------------

                                umb_insert:
000037F9 1E                         push    ds

                                    ; 31/12/2022
                                    ; ds = cs

                                    ;mov    ds,[cs:DevDOSData]
000037FA 8E1E[6024]                 mov     ds,[DevDOSData] ; 31/12/2022
                                    ;mov    ds,[8Ch]
000037FE 8E1E8C00                   mov     ds,[UMB_ARENA]          ; es = UMB_HEAD
00003802 8CD8                       mov     ax,ds
00003804 8EC0                       mov     es,ax
                                ui_next:
00003806 39D8                       cmp     ax,bx                   ; Q: is current block above
                                                                    ;    new block
00003808 770F                       ja      short ui_insert             ; Y: insert it
                                                                    ; Q: is current block the
                                                                    ;    last
0000380A 26803E00005A               cmp     byte [es:ARENA.SIGNATURE],arena_signature_end ; 'Z'
00003810 745C                       je      short ui_append             ; Y: append new block to chain
                                                                    ; N: get next block
00003812 8ED8                       mov     ds,ax                   ; M005
                                    ;call   get_next                ; ax = es = next block
00003814 E83B01                     call    _get_next_ ; 13/04/2019 - Retro DOS v4.0
00003817 EBED                       jmp     short ui_next

                                ui_insert:
00003819 8CD9                       mov     cx,ds                   ; ds = previous arena
0000381B 41                         inc     cx                      ; top of previous block

0000381C 29D9                       sub     cx,bx
0000381E F7D9                       neg     cx                      ; cx = size of used block
                                    ;mov    byte [0],'M'
00003820 C60600004D                 mov     byte [ARENA.SIGNATURE],arena_signature_normal ; 'M'
                                    ;mov    word [1],8
00003825 C70601000800               mov     word [ARENA.OWNER],8  ; mark as system owned
                                    ;mov    [3],cx
0000382B 890E0300                   mov     [ARENA.SIZE],cx
                                    ;mov    word [8],4353h ; 'SC'
0000382F C70608005343               mov     word [ARENA.NAME],'SC' ; 4353h

                                ; prepare the arena at start of new block

00003835 8EC3                       mov     es,bx
00003837 26C60600004D               mov     byte [es:ARENA.SIGNATURE],arena_signature_normal ; 'M'
0000383D 26C70601000000             mov     word [es:ARENA.OWNER],arena_owner_system ; 0
                                                                    ; mark as free
00003844 83EA02                     sub     dx,2                    ; make room for arena at
                                                                    ; start & end of new block
00003847 2689160300                 mov     [es:ARENA.SIZE],dx

                                ; prepare arena at end of new block

0000384C 01D3                       add     bx,dx
0000384E 43                         inc     bx
0000384F 8EC3                       mov     es,bx                   ; es=arena at top of new block
00003851 43                         inc     bx                      ; bx=top of new block

                                                                    ; ax contains arena just above
                                                                    ; this block
00003852 29D8                       sub     ax,bx                   ; ax = size of used block

00003854 26C60600004D               mov     byte [es:ARENA.SIGNATURE],arena_signature_normal
0000385A 26C70601000800             mov     word [es:ARENA.OWNER],8       ; mark as system owned
00003861 26A30300                   mov     [es:ARENA.SIZE],ax
00003865 26C70608005343             mov     word [es:ARENA.NAME],'SC' ; 4353h

0000386C EB47                       jmp     short ui_done

                                ui_append:
                                                                    ; es = arena of last block
0000386E 2603060300                 add     ax,[es:ARENA.SIZE]    ; ax=top of last block-1 para
00003873 26832E030001               sub     word [es:ARENA.SIZE],1; reflect the space we are
                                                                    ; going to rsrv on top of this
                                                                    ; block for the next arena.
                                    ; 13/05/2019
00003879 26C60600004D               mov     byte [es:ARENA.SIGNATURE],arena_signature_normal

0000387F 89C1                       mov     cx,ax                   ; cx=top of prev block-1
00003881 40                         inc     ax
00003882 29D8                       sub     ax,bx                   ; ax=top of prev block -
                                                                    ;    seg. address of new block
00003884 F7D8                       neg     ax

00003886 8EC1                       mov     es,cx                   ; ds = arena of unused block

00003888 26C60600004D               mov     byte [es:ARENA.SIGNATURE],arena_signature_normal
0000388E 26C70601000800             mov     word [es:ARENA.OWNER],8       ; mark as system owned
00003895 26A30300                   mov     [es:ARENA.SIZE],ax
00003899 26C70608005343             mov     word [es:ARENA.NAME],'SC'

                                ; prepare the arena at start of new block
000038A0 8EC3                       mov     es,bx
000038A2 26C60600005A               mov     byte [es:ARENA.SIGNATURE],arena_signature_end
000038A8 26C70601000000             mov     word [es:ARENA.OWNER],arena_owner_system
                                                                    ; mark as free
000038AF 4A                         dec     dx                      ; make room for arena
000038B0 2689160300                 mov     [es:ARENA.SIZE],dx
                                ui_done:
                                uc_done: ; 31/12/2022 ; *!
000038B5 1F                         pop     ds
                                    ; ds = cs ; 31/12/2022
                                ;uc_done:   ; 18/12/2022
                                au_exit:    ; 09/09/2023
000038B6 C3                         retn

                                ;--------------------------------------------------------------------------
```

```
                                ;
                                ; procedure : AllocUMB
                                ;
                                ;   Allocate all UMBs and link it to DOS arena chain
                                ;
                                ;--------------------------------------------------------------------------

                                AllocUMB:
                                    ; 31/12/2022
                                    ; ds = cs
000038B7 E84700                     call    InitAllocUMB            ; link in the first UMB
000038BA 72FA                       jc      short au_exit           ; quit on error
                                au_next:
000038BC E87000                     call    umb_allocate            ; allocate
000038BF 7205                       jc      short au_coalesce
000038C1 E835FF                     call    umb_insert              ; & insert till no UMBs
000038C4 EBF6                       jmp     short au_next
                                au_coalesce:
                                    ; 09/09/2023
                                ;   call    umb_coalesce            ; coalesce all UMBs
                                ;au_exit:
                                ;   ; 31/12/2022
                                ;   ; ds = cs
                                ;   retn

                                    ; 09/09/2023
                                    ;jmp    short umb_coalesce

                                ; 09/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)

                                ; 13/04/2019 - Retro DOS v4.0

                                ;--------------------------------------------------------------------------
                                ;
                                ;** umb_coalesce - Combine free blocks ahead with current block
                                ;
                                ;   Coalesce adds the block following the argument to the argument block,
                                ;   if it's free. Coalesce is usually used to join free blocks, but
                                ;   some callers (such as $setblock) use it to join a free block to it's
                                ;   preceeding allocated block.
                                ;
                                ;   EXIT    'C' clear if OK
                                ;             (ds) unchanged, this block updated
                                ;             (ax) = address of next block, IF not at end
                                ;           'C' set if arena trashed
                                ;   USES    cx, di, ds, es
                                ;
                                ;--------------------------------------------------------------------------

                                umb_coalesce:
                                    ; 31/12/2022
                                    ; ds = cs
000038C6 1E                         push    ds ; *!

000038C7 31FF                       xor     di, di

                                    ;mov    es,[cs:DevDOSData]
                                    ; 31/12/2022
000038C9 8E06[6024]                 mov     es,[DevDOSData]
000038CD 268E068C00                 mov     es,[es:UMB_ARENA]       ; es = UMB_HEAD
                                uc_nextfree:
000038D2 8CC0                       mov     ax,es
000038D4 8ED8                       mov     ds,ax
                                    ;cmp    [es:1],di
000038D6 26393E0100                 cmp     [es:ARENA.OWNER],di     ; Q: is current arena free
000038DB 7407                       je      short uc_again          ; Y: try to coalesce with next block
                                                                    ; N: get next arena
000038DD E86B00                     call    get_next                ; es, ax = next arena
000038E0 72D3                       jc      short uc_done  ; *!
000038E2 EBEE                       jmp     short uc_nextfree
                                uc_again:
000038E4 E86400                     call    get_next                ; es, ax = next arena
000038E7 72CC                       jc      short uc_done  ; *!
                                uc_check:
000038E9 26393E0100                 cmp     [es:ARENA.OWNER],di     ; Q: is arena free
000038EE 75E2                       jne     short uc_nextfree       ; N: get next free arena
                                                                    ; Y: coalesce
000038F0 268B0E0300                 mov     cx,[es:ARENA.SIZE]        ; cx <- next block size
000038F5 41                         inc     cx                        ; cx <- cx + 1 (for header size)
                                    ;add    [3],cx
000038F6 010E0300                   add     [ARENA.SIZE],cx             ; current size <- current size + cx
000038FA 268A0D                     mov     cl,[es:di]                ; move up signature
000038FD 880D                       mov     [di],cl
000038FF EBE3                       jmp     short uc_again          ; try again

                                    ; 18/12/2022
                                ;uc_done:
                                    ;retn

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : InitAllocUMB
                                ;
                                ;--------------------------------------------------------------------------

                                InitAllocUMB:
                                    ; 31/12/2022
                                    ; ds = cs
00003901 E8D6D2                     call    IsXMSLoaded
00003904 7527                       jnz     short iau_err           ; quit on no XMS driver
00003906 B452                       mov     ah,52h
00003908 CD21                       int     21h                     ; get DOS DATA seg
                                    ; 31/12/2022
                                    ; ds = cs
                                    ;mov    [cs:DevDOSData],es      ; & save it for later
0000390A 8C06[6024]                 mov     [DevDOSData],es               ; & save it for later
0000390E B81043                     mov     ax,4310h
00003911 CD2F                       int     2Fh
                                    ;mov    [cs:DevXMSAddr],bx      ; get XMS driver address
                                    ;mov    [cs:DevXMSAddr+2],es
00003913 891E[4924]                 mov     [DevXMSAddr],bx               ; get XMS driver address
00003917 8C06[4B24]                 mov     [DevXMSAddr+2],es
                                    ; 31/12/2022
0000391B 803E[5F24]00               cmp     byte [FirstUMBLinked],0
                                    ;cmp    byte [cs:FirstUMBLinked],0 ; have we already linked a UMB?
                                    ;jne    short ia_1              ; quit if we already did it
                                    ; 12/12/2022
00003920 770A                       ja      short ia_1 ; cf=0
00003922 E83900                     call    LinkFirstUMB            ; else link the first UMB
                                    ;jc     short iau_err
                                    ; 12/12/2022
00003925 7207                       jc      short iau_err2  ; cf=1
                                    ; 31/12/2022
                                    ; ds = cs
```

```
41009 00003927 C606[5F24]FF             mov     byte [FirstUMBLinked],0FFh ; mark that 1st UMB linked
41010                                   ;mov    byte [cs:FirstUMBLinked],0FFh ; mark that 1st UMB linked
41011                                 ia_1:
41012                                   ; 12/12/2022
41013                                   ; cf=0
41014                                   ;clc
41015 0000392C C3                       retn
41016                                 iau_err:
41017 0000392D F9                       stc
41018                                 iau_err2:
41019 0000392E C3                       retn
41020
41021                                 ;--------------------------------------------------------------------------
41022                                 ;
41023                                 ; Procedure Name   : umb_allocate
41024                                 ;
41025                                 ; Inputs           : DS = data
41026                                 ;
41027                                 ; Outputs          : if UMB available
41028                                 ;                          Allocates the largest available UMB and
41029                                 ;                          BX = segment of allocated block
41030                                 ;                          DX = size of allocated block
41031                                 ;                          NC
41032                                 ;                    else
41033                                 ;                          CY
41034                                 ;
41035                                 ; Uses                     : BX, DX
41036                                 ;
41037                                 ;--------------------------------------------------------------------------
41038
41039                                 umb_allocate:
41040                                   ; 31/12/2022
41041                                   ; ds = cs
41042 0000392F 50                       push    ax
41043 00003930 B410                     mov     ah,XMM_REQUEST_UMB ; 16
41044 00003932 BAFFFF                   mov     dx,0FFFFh               ; try to allocate largest
41045                                                                   ;   possible
41046                                   ; 31/12/2022
41047 00003935 FF1E[4924]               call    far [DevXMSAddr]
41048                                   ;call   far [cs:DevXMSAddr]
41049                                                                   ; dx now contains the size of
41050                                                                   ; the largest UMB
41051 00003939 09D2                     or      dx,dx
41052 0000393B 740B                     jz      short ua_err
41053
41054 0000393D B410                     mov     ah,XMM_REQUEST_UMB ; 16
41055
41056                                   ; 31/12/2022
41057 0000393F FF1E[4924]               call    far [DevXMSAddr]
41058                                   ;call   far [cs:DevXMSAddr]
41059
41060 00003943 83F801                   cmp     ax,1                    ; Q: was the reqst successful
41061                                   ;jne    short ua_err            ; N: error
41062                                   ; 27/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
41063 00003946 7601                     jna     short ua_done ; if ax=1 then cf=0, else cf=1 (ax=0)
41064                                 ua_err:
41065 00003948 F9                       stc
41066
41067                                   ;clc
41068                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
41069                                   ; 12/12/2022
41070                                   ; cf=0
41071                                   ;clc
41072                                 ua_done:
41073 00003949 58                       pop     ax
41074 0000394A C3                       retn
41075                                   ; 27/07/2023
41076                                 ;ua_err:
41077                                   ;stc
41078                                   ;jmp    short ua_done
41079
41080                                 ;--------------------------------------------------------------------------
41081                                 ;
41082                                 ;** get_next - Find Next item in Arena
41083                                 ;
41084                                 ;   ENTRY   dS - pointer to block head
41085                                 ;   EXIT    AX,ES - pointers to next head
41086                                 ;           'C' set if arena damaged
41087                                 ;
41088                                 ;--------------------------------------------------------------------------
41089
41090                                   ; 01/11/2022
41091                                 get_next:
41092 0000394B 803E00005A               cmp     byte [0],arena_signature_end ; 'Z'
41093 00003950 740A                     je      short gn_err
41094                                 _get_next_:
41095 00003952 8CD8                     mov     ax,ds                   ; ax=current block
41096 00003954 03060300                 add     ax,[ARENA.SIZE]                 ; ax=ax + current block length
41097 00003958 40                       inc     ax                      ; remember that header!
41098 00003959 8EC0                     mov     es,ax
41099                                   ;clc
41100                                   ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
41101                                   ; 11/12/2022
41102                                   ; cf=0
41103                                   ;clc
41104 0000395B C3                       retn
41105                                 gn_err:
41106 0000395C F9                       stc
41107                                   ; 11/12/2022
41108                                 lfu_err:    ; cf=1
41109 0000395D C3                       retn
41110
41111                                 ;--------------------------------------------------------------------------
41112                                 ;
41113                                 ; procedure : LinkFirstUMB
41114                                 ;
41115                                 ;--------------------------------------------------------------------------
41116
41117                                   ; 01/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
41118                                   ; (SYSINIT:2F81h)
41119                                 LinkFirstUMB:
41120                                   ; 31/12/2022
41121                                   ; ds = cs
41122 0000395E E8CEFF                   call    umb_allocate
41123 00003961 72FA                     jc      short lfu_err  ; ds = cs ; 31/12/2022
41124
41125                                 ; bx = segment of allocated UMB
41126                                 ; dx = size of UMB
41127
41128                                   ; 31/12/2022
41129                                   ; ds = cs
41130
41131 00003963 CD12                     int     12h                     ; ax = size of memory
41132 00003965 B106                     mov     cl,6
```

```
00003967 D3E0                       shl     ax,cl                   ; ax = size in paragraphs
                                    
00003969 89C1                       mov     cx,ax                   ; cx = size in paras
0000396B 29D8                       sub     ax,bx                   ; ax = - size of unused block
                                    
0000396D F7D8                       neg     ax
                                    
                                    ;sub    cx,1                    ; cx = first umb_arena
                                    ; 09/09/2023
0000396F 49                         dec     cx
00003970 8EC1                       mov     es,cx                   ; es = first umb_arena
                                    
00003972 26C60600004D               mov     byte [es:ARENA.SIGNATURE],arena_signature_normal ; 'M'
00003978 26C70601000800             mov     word [es:ARENA.OWNER],8       ; mark as system owned
                                    
0000397F 26A30300                   mov     [es:ARENA.SIZE],ax
00003983 26C70608005343             mov     word [es:ARENA.NAME],'SC' ; 4353h
                                    
                                  ; put in the arena for the first UMB
                                    
0000398A 8EC3                       mov     es,bx                   ; es has first free umb seg
0000398C 26C60600005A               mov     byte [es:ARENA.SIGNATURE],arena_signature_end ; 'Z'
00003992 26C70601000000             mov     word [es:ARENA.OWNER],arena_owner_system ; 0
                                                                    ; mark as free
00003999 4A                         dec     dx                      ; make room for arena
0000399A 2689160300                 mov     [es:ARENA.SIZE],dx
                                    
                                    ;mov    es,[cs:DevDOSData]
                                    ; 31/12/2022
0000399F 8E06[6024]                 mov     es,[DevDOSData] ; ds = cs
000039A3 BF8C00                     mov     di,UMB_ARENA ; 8Ch
000039A6 26890D                     mov     [es:di],cx              ; initialize umb_head in DOS
                                                                    ;  data segment with the arena
                                                                    ;  just below Top of Mem
                                    
                                  ; we must now scan the arena chain and update the size of the last arena
                                    
000039A9 BF2400                     mov     di,DOS_ARENA ; 24h
000039AC 268E05                     mov     es,[es:di]              ; es = start arena
000039AF 31FF                       xor     di,di
                                  ;scan_next
                                  ; 09/12/2022
                                  scannext:
000039B1 26803D5A                   cmp     byte [es:di],arena_signature_end  ; 'Z'
000039B5 740C                       je      short got_last
                                    
000039B7 8CC0                       mov     ax,es
000039B9 2603060300                 add     ax,[es:ARENA.SIZE]
000039BE 40                         inc     ax
000039BF 8EC0                       mov     es,ax
                                    ;jmp    short scan_next
                                    ; 09/12/2022
000039C1 EBEE                       jmp     short scannext
                                  got_last:
                                    ;sub    word [es:ARENA.SIZE],1
                                    ; 09/09/2023
000039C3 26FF0E0300                 dec     word [es:ARENA.SIZE]
                                    
000039C8 26C60600004D               mov     byte [es:ARENA.SIGNATURE],arena_signature_normal ; 'M'
                                    ;clc
                                    ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ; 11/12/2022
                                    ; cf=0
                                    ;clc
000039CE C3                         retn
                                    
                                  ; 11/12/2022
                                  ;;lfu_err:
                                  ;   ;stc
                                  ;   ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                  ;   ; 11/12/2022
                                  ;   ; cf=1
                                  ;   ;stc
                                  ;   retn
                                    
                                  ;--------------------------------------------------------------------------
                                  ;
                                  ; procedure : ShrinkUMB
                                  ;
                                  ;   Shrinks the current UMB in use, so that the unused portions
                                  ;   of the UMB is given back to the DOS free mem pool
                                  ;
                                  ;--------------------------------------------------------------------------
                                    
                                  ShrinkUMB:
                                    ; 12/12/2022
                                    ; ds = cs
000039CF 833E[4324]00               cmp     word [DevUMBAddr],0
                                    ;cmp    word [cs:DevUMBAddr],0
000039D4 741F                       je      short su_exit
000039D6 06                         push    es
                                    ; 01/01/2023
                                    ;push   bx
                                    ; 12/12/2022
                                    ;mov    bx,[cs:DevUMBFree]
                                    ;sub    bx,[cs:DevUMBAddr]
                                    ;mov    es,[cs:DevUMBAddr]
000039D7 8B1E[4724]                 mov     bx,[DevUMBFree]
000039DB 2B1E[4324]                 sub     bx,[DevUMBAddr]
000039DF 8E06[4324]                 mov     es,[DevUMBAddr]
                                    
000039E3 B8004A                     mov     ax,4A00h
000039E6 CD21                       int     21h
                                            ; DOS - 2+ - ADJUST MEMORY BLOCK SIZE (SETBLOCK)
                                            ; ES = segment address of block to change
                                            ; BX = new size in paragraphs
000039E8 8CC0                       mov     ax,es
000039EA 48                         dec     ax
000039EB 8EC0                       mov     es,ax
000039ED 26C70601000800             mov     word [es:ARENA.OWNER],8
                                    ; 01/01/2023
                                    ;pop    bx
000039F4 07                         pop     es
                                  su_exit:
000039F5 C3                         retn
                                    
                                  ;--------------------------------------------------------------------------
                                  ;
                                  ; procedure : UnlinkUMB
                                  ;
                                  ;   Unlinks the UMBs from the DOS arena chain
                                  ;
                                  ;--------------------------------------------------------------------------
```

```
                                UnlinkUMB:
                                    ; 12/12/2022
                                    ; ds = cs
000039F6 1E                         push    ds
000039F7 06                         push    es
                                    ; 12/12/2022
000039F8 803E[5F24]00               cmp     byte [FirstUMBLinked],0
                                    ;cmp    byte [cs:FirstUMBLinked],0
000039FD 7420                       je      short ulu_x             ; nothing to unlink
                                    ; 12/12/2022
000039FF 8E06[6024]                 mov     es,[DevDOSData]
                                    ;mov    es,[cs:DevDOSData]      ; get DOS data seg
00003A03 268E1E2400                 mov     ds,[es:DOS_ARENA]
00003A08 268B3E8C00                 mov     di,[es:UMB_ARENA]
                                ulu_next:
00003A0D E83BFF                     call    get_next
00003A10 720D                       jc      short ulu_x
00003A12 39C7                       cmp     di,ax                   ; is the next one UMB ?
00003A14 7404                       je      short ulu_found
00003A16 8ED8                       mov     ds,ax
00003A18 EBF3                       jmp     short ulu_next
                                ulu_found:
                                    ;mov    byte [0],'Z'
00003A1A C60600005A                 mov     byte [ARENA.SIGNATURE],arena_signature_end ; 'Z'
                                ulu_x:
00003A1F 07                         pop     es
00003A20 1F                         pop     ds
00003A21 C3                         retn

                                ; ----------------------------------------------------------------------------
                                ; SYSINIT2.ASM - MSDOS 6.0 - 1991
                                ; ----------------------------------------------------------------------------
                                ; 14/04/2019 - Retro DOS v4.0

                                ; Multiple configuration block support  Created 16-Mar-1992 by JeffPar
                                ;
                                ; Summary:
                                ;
                                ;   The procedure "organize" crunches the in-memory copy of config.sys
                                ;   into lines delimited by CR/LF (sometimes no CR, but *always* an LF)
                                ;   with the leading "keyword=" replaced by single character codes (eg, B
                                ;   for BUFFERS, D for DEVICE, Z for any unrecognized keyword); see comtab
                                ;   and/or config.inc for the full list.
                                ;
                                ;   [blockname] and INCLUDE are the major syntactical additions for multi-
                                ;   configuration support. blockname is either MENU, which contains one
                                ;   or more MENUITEM lines, an optional MENUDEFAULT (which includes optional
                                ;   time-out), or any user-defined keyword, such as NETWORK, CD-ROM, etc.
                                ;   INCLUDE allows the current block to name another block for inclusion
                                ;   during the processing phase of CONFIG.SYS. An INCLUDE is only honored
                                ;   once, precluding nasty infinite-loop scenarios. If blocks are present
                                ;   without a MENU block, then only lines inside COMMON blocks are processed.
                                ;
                                ; Example:
                                ;
                                ;   [menu]
                                ;   menuitem=misc,Miscellaneous
                                ;   menuitem=network,Network Configuration
                                ;   menudefault=network,15
                                ;
                                ;   [network]
                                ;   include misc
                                ;   device=foo
                                ;
                                ;   [misc]
                                ;   device=bar
                                ;   include alternate
                                ;
                                ;   [alternate]
                                ;   device=tar
                                ;
                                ;
                                ;   When the menu is displayed
                                ;
                                ;    1. Miscellaneous
                                ;    2. Network Configuration
                                ;
                                ;   #2 is highlighted as the default option, and will be automatically
                                ;   selected after 15 seconds. It will invoke the following lines in the
                                ;   following order:
                                ;
                                ;       DEVICE=BAR
                                ;       DEVICE=TAR
                                ;       DEVICE=FOO
                                ;

                                ;MULTI_CONFIG equ 1

                                ; the following depend on the positions of the various letters in switchlist

                                switchnum   equ 11111000b ; 0F8h  ; which switches require number

                                flagec35    equ 00000100b ; 4     ; electrically compatible 3.5 inch disk drive
                                flagdrive   equ 00001000b ; 8
                                flagcyln    equ 00010000b ; 16
                                flagseclim  equ 00100000b ; 32
                                flagheads   equ 01000000b ; 64
                                flagff              equ 10000000b ; 128

                                ;--------------------------------------------------------------------------
                                ; 19/04/2019 - Retro DOS v4.0

                                ; MSDOS 6.21 IO.SYS - SYSINIT:3E78h

                                ; 01/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ; MSDOS 5.0 IO.SYS - SYSINIT:3054h

00003A22 00                     insert_blank:       db      0

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : setparms
                                ;
                                ; the following set of routines is used to parse the drivparm = command in
                                ; the config.sys file to change the default drive parameters.
                                ;
                                ;--------------------------------------------------------------------------

                                setparms:
00003A23 1E                         push    ds
00003A24 50                         push    ax
00003A25 53                         push    bx
00003A26 51                         push    cx
00003A27 52                         push    dx
```

```
                                
00003A28 0E                         push    cs
00003A29 1F                         pop     ds
                                
00003A2A 31DB                       xor     bx,bx
00003A2C 8A1E[1C4F]                 mov     bl,[drive]
                                    ; 18/12/2022
00003A30 43                         inc     bx
                                    ;inc    bl                      ; get it correct for ioctl call
                                                                    ; (1=a,2=b...)
00003A31 BA[BE4D]                   mov     dx,deviceparameters
                                    ;mov    ah,IOCTL ; 44h
                                    ;mov    al,GENERIC_IOCTL ; 0Dh
                                    ; 04/07/2023
00003A34 B80D44                     mov     ax,(IOCTL<<8)|GENERIC_IOCTL
                                    ;mov    ch,RAWIO ; 8
                                    ;mov    cl,SET_DEVICE_PARAMETERS ; 40h
                                    ; 04/07/2023
00003A37 B94008                     mov     cx,(RAWIO<<8)|SET_DEVICE_PARAMETERS
00003A3A CD21                       int     21h
                                
                                ; 27/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
00003A3C 8A26[1D4F]                 mov     ah,[switches]
                                    ;mov    al,[deviceparameters+20]
00003A40 A0[D24D]                   mov     al,[deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERTRACK]
00003A43 8A0E[1C4F]                 mov     cl,[drive]
                                ;
                                ;; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;   ;;mov   ax,Bios_Data          ; get Bios_Data segment
                                ;   ;mov    ax,KERNEL_SEGMENT ; 70h
                                ;   ; 21/10/2022
                                ;   ;mov    ax,DOSBIODATASEG ; 0070h
                                ;   ;mov    ds,ax                   ; set Bios_Data segment
                                ;
                                ;   ; 27/07/2023
                                ;   ;;test  word [cs:switches],flagec35 ; 4
                                ;   ;test   byte [cs:switches],flagec35
                                ;   ;jz     short not_ec35
                                ;
                                ;   ; 27/07/2023
                                ;   ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;   ;;test  word [switches],flagec35 ; 4
                                ;   ; 12/12/2022
                                ;   ;test   byte [switches],flagec35 ; 4
                                ;   ;jz     short eot_ok
                                ;
                                    ;mov    cl,[cs:drive]           ; which drive was this for?
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;mov    cl,[drive]
                                    ; 27/07/2023
                                    ;mov    ax,DOSBIODATASEG ; 0070h
                                    ;mov    ds,ax
                                
00003A47 BA7000                     mov     dx,DOSBIODATASEG
00003A4A 8EDA                       mov     ds,dx
                                
00003A4C F6C404                     test    ah,flagec35     ; test byte [cs:switches],flagec35
00003A4F 7408                       jz      short not_ec35
                                
                                    ;mov    al,1                    ; assume drive 0
                                    ;shl    al,cl                   ; set proper bit depending on drive
                                    ;;or    [531h],al ; (MSDOS 6.21 IO.SYS Offset SYINIT:3EACh)
                                    ;or     [ec35_flag],al          ; set the bit in the permanent flags
                                    ; 27/07/2023
00003A51 B401                       mov     ah,1
00003A53 D2E4                       shl     ah,cl
00003A55 0826[A204]                 or      [ec35_flag],ah
                                
                                ; 07/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                ;   MSDOS 6.21 IO.SYS - SYINIT:3EB0h
                                ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                not_ec35:
                                ;   Now adjust the BIOS's EOT variable if our new drive has more
                                ;   sectors per track than any old ones.
                                
                                    ; 27/07/2023
                                    ;;mov   al,[cs:deviceparameters+20]
                                    ;mov    al,[cs:deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERTRACK]
                                
                                    ;cmp    al,[12Ch] ; (MSDOS 6.21 IO.SYS Offset SYINIT:3EB4h)
00003A59 3A06[2C01]                 cmp     al,[eot]
00003A5D 7603                       jbe     short eot_ok
00003A5F A2[2C01]                   mov     [eot],al
                                eot_ok:
00003A62 5A                         pop     dx                      ; fix up all the registers
00003A63 59                         pop     cx
00003A64 5B                         pop     bx
00003A65 58                         pop     ax
00003A66 1F                         pop     ds ; 13/05/2019
00003A67 C3                         retn
                                
                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : diddleback
                                ;
                                ; replace default values for further drivparm commands
                                ;
                                ;--------------------------------------------------------------------------
                                
                                diddleback:
00003A68 1E                         push    ds
00003A69 0E                         push    cs
00003A6A 1F                         pop     ds
                                    ;mov    word [deviceparameters+4],80
00003A6B C706[C24D]5000             mov     word [deviceparameters+A_DEVICEPARAMETERS.DP_CYLINDERS],80
                                    ;mov    byte [deviceparameters+1],2
00003A71 C606[BF4D]02               mov     byte [deviceparameters+A_DEVICEPARAMETERS.DP_DEVICETYPE],DEV_3INCH720KB ; 2
                                    ;mov    word [deviceparameters+2],0
00003A76 C706[C04D]0000             mov     word [deviceparameters+A_DEVICEPARAMETERS.DP_DEVICEATTRIBUTES],0
00003A7C C706[1D4F]0000             mov     word [switches],0       ; zero all switches
00003A82 1F                         pop     ds
00003A83 C3                         retn
                                
                                ; 03/01/2023
                                %if 0
                                
                                ; 15/04/2019 - Retro DOS v4.0
                                
                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : parseline
                                ;
                                ; entry point is parseline. al contains the first character in command line.
                                ;
```

```
                        ;--------------------------------------------------------------------------

                            ; 03/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
                            ; (SYSINIT:3EDFh)

                            ; 01/11/2022 - Retrodos v4.0 (Modified MSDOS 5.0 IO.SYS)
                            ; (SYSINIT:30ACh)
                        parseline:
                            ; 03/01/2023
                            ; ds = cs ; *

                            ;push  ds ; *

                            ;push  cs ; *
                            ;pop   ds ; *

                        nextswtch:
                            cmp    al,cr                ; carriage return?
                            je     short done_line
                            cmp    al,lf                ; linefeed?
                            je     short put_back       ; put it back and done

                        ; anything less or equal to a space is ignored.

                            cmp    al,' '                 ; space?
                            jbe    short getnext        ; skip over space
                            cmp    al,'/'
                            je     short getparm
                            stc                         ; mark error invalid-character-in-input
                            ;jmp   short exitpl
                            ; 03/01/2023
                        swterr:
                            retn

                        getparm:
                            call   check_switch
                            mov    [switches],bx        ; save switches read so far
                            jc     short swterr
                        getnext:
                            call   getchr
                            ;jc    short done_line
                            ;jmp   short nextswtch
                            ; 03/01/2023
                            jnc    short nextswtch
                        ;swterr:
                            ;jmp   short exitpl          ; exit if error

                        done_line:
                            ; 12/12/2022
                            test   byte [switches],flagdrive ; 8
                            ;test  word [switches],flagdrive ; 8 ; see if drive specified
                            jnz    short okay
                            stc                         ; mark error no-drive-specified
                            ;jmp   short exitpl
                            ; 03/01/2023
                            retn

                        okay:
                            mov    ax,[switches]
                            and    ax,0003h           ; get flag bits for changeline and non-rem
                            mov    [deviceparameters+A_DEVICEPARAMETERS.DP_DEVICEATTRIBUTES],ax
                            mov    word [deviceparameters+A_DEVICEPARAMETERS.DP_TRACKTABLEENTRIES],0
                            ;clc                        ; everything is fine
                            ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                            ; 12/12/2022
                            ; cf=0
                            ;clc
                            ;call  setdeviceparameters
                            ; 03/01/2023
                            jmp    setdeviceparameters
                        ;exitpl:
                            ; 03/01/2023
                            ; ds = cs
                            ;pop   ds ; *
                            retn
                        put_back:
                            inc    word [count]         ; one more char to scan
                            dec    word [chrptr]        ; back up over linefeed
                            jmp    short done_line

                        %endif

                        ;--------------------------------------------------------------------------
                        ;
                        ; procedure : check_switch
                        ;
                        ; processes a switch in the input. it ensures that the switch is valid, and
                        ; gets the number, if any required, following the switch. the switch and the
                        ; number *must* be separated by a colon. carry is set if there is any kind of
                        ; error.
                        ;
                        ;--------------------------------------------------------------------------

                        ; 09/09/2023

                        err_swtch:
                            xor    bx,cx                ; remove this switch from the records
                        err_check:
                            stc
                        err_chk:
                        done_swtch: ; 09/09/2023 (cf=0)
                            retn

                            ; 09/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)

                        check_switch:
                            call   getchr
                            ;jc    short err_check
                            jc     short err_chk
                               and     al,0DFh                 ; convert it to upper case
                            cmp    al,'A'
                            ;jb    short err_check
                            jb     short err_chk ; 15/04/2019 - Retro DOS v4.0
                            cmp    al,'Z'
                            ja     short err_check

                            push   es

                            push   cs
                            pop    es

                            ;mov   cl,[switchlist]              ; get number of valid switches
                            ;mov   ch,0
                            ;mov   di,1+switchlist              ; point to string of valid switches
```

```
                                        ; 09/09/2023
00003A9A BF[4250]                       mov     di,switchlist
00003A9D 8A0D                           mov     cl,[di]
00003A9F B500                           mov     ch,0
00003AA1 47                             inc     di      ; 1+switchlist

00003AA2 F2AE                           repne   scasb

00003AA4 07                             pop     es
00003AA5 75DF                           jnz     short err_check

00003AA7 B80100                         mov     ax,1
00003AAA D3E0                           shl     ax,cl                   ; set bit to indicate switch
00003AAC 8B1E[1D4F]                     mov     bx,[switches]           ; get switches so far
00003AB0 09C3                           or      bx,ax                   ; save this with other switches
00003AB2 89C1                           mov     cx,ax
                                        ; 12/12/2022
00003AB4 A8F8                           test    al,switchnum ; 0F8h
                                        ;test   ax,switchnum ; 0F8h     ; test against switches that require number to follow
00003AB6 74CF                           jz      short done_swtch

00003AB8 E8F40C                         call    getchr
00003ABB 72C7                           jc      short err_swtch

00003ABD 3C3A                           cmp     al,':'
00003ABF 75C3                           jne     short err_swtch

00003AC1 E8EB0C                         call    getchr
00003AC4 53                             push    bx                      ; preserve switches
                                        ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                        ;mov    byte [cs:sepchr],' '    ; allow space separators
                                        ; 12/12/2022
                                        ; ds = cs
00003AC5 C606[AE02]20                   mov     byte [sepchr],' '
00003ACA E8980D                         call    getnum
                                        ;mov    byte [cs:sepchr],0
                                        ; 12/12/2022
00003ACD C606[AE02]00                   mov     byte [sepchr],0
00003AD2 5B                             pop     bx                      ; restore switches

                                ; because getnum does not consider carriage-return or line-feed as ok, we do
                                ; not check for carry set here. if there is an error, it will be detected
                                ; further on (hopefully).

                                    ; 09/09/2023
                                    ;call   process_num
                                    ;jmp    short process_num

                                ;done_swtch:
                                ;   ;clc
                                ;   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;   ; 12/12/2022
                                ;   ; cf=0
                                ;   ;clc
                                ;   retn

                                ;-----------------------------------------------------------------------------
                                ;
                                ; procedure : process_num
                                ;
                                ; this routine takes the switch just input, and the number following (if any),
                                ; and sets the value in the appropriate variable. if the number input is zero
                                ; then it does nothing - it assumes the default value that is present in the
                                ; variable at the beginning. zero is ok for form factor and drive, however.
                                ;
                                ;-----------------------------------------------------------------------------

                                    ; 02/11/2022 - Retrodos v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:3156h)
                                process_num:
00003AD3 850E[1D4F]                     test    [switches],cx           ; if this switch has been done before,
00003AD7 752B                           jnz     short done_ret          ; ignore this one.
                                        ; 12/12/2022
00003AD9 F6C108                         test    cl,flagdrive ; 8
                                        ;test   cx,flagdrive ; 8
00003ADC 7404                           jz      short try_f
00003ADE A2[1C4F]                       mov     byte [drive],al
                                        ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                        ;jmp    short done_ret
                                        ; 12/12/2022
                                        ; cf=0
00003AE1 C3                             retn    ; 13/05/2019
                                try_f:
                                        ; 12/12/2022
00003AE2 F6C180                         test    cl,flagff ; 80h
                                        ;test   cx,flagff ; 80h
00003AE5 7404                           jz      short try_t

                                ; ensure that we do not get bogus form factors that are not supported

                                        ;mov    [deviceparameters+1],al
00003AE7 A2[BF4D]                       mov     [deviceparameters+A_DEVICEPARAMETERS.DP_DEVICETYPE],al
                                        ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                        ;jmp    short done_ret
                                        ; 12/12/2022
                                        ; cf=0
00003AEA C3                             retn    ; 13/05/2019
                                try_t:
00003AEB 09C0                           or      ax,ax
00003AED 7415                           jz      short done_ret          ; if number entered was 0, assume default value
                                        ; 12/12/2022
00003AEF F6C110                         test    cl,flagcyln ; 10h
                                        ;test   cx,flagcyln ; 10h
00003AF2 7404                           jz      short try_s

                                        ;mov    [deviceparameters+4],ax
00003AF4 A3[C24D]                       mov     [deviceparameters+A_DEVICEPARAMETERS.DP_CYLINDERS],ax
                                        ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                        ;jmp    short done_ret
                                        ; 12/12/2022
                                        ; cf=0
00003AF7 C3                             retn    ; 13/05/2019
                                try_s:
                                        ; 12/12/2022
00003AF8 F6C120                         test    cl,flagseclim ; 20h
                                        ;test   cx,flagseclim ; 20h
00003AFB 7404                           jz      short try_h
00003AFD A3[1A4F]                       mov     [slim],ax
                                        ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                        ;jmp    short done_ret
                                        ; 12/12/2022
                                        ; cf=0
00003B00 C3                             retn    ; 13/05/2019
```

```
                                    ; must be for number of heads
                                
                                    try_h:
00003B01 A3[184F]                       mov     [hlim],ax
                                    done_ret:
                                        ;clc
                                        ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                        ; 12/12/2022
                                        ; cf=0 (test instruction resets cf)
                                        ;clc
00003B04 C3                             retn
                                
                                
                                    ; 16/04/2024 - Retro DOS v5.0
                                    ; 03/01/2023 - Retro DOS v4.2
                                    %if 1
                                
                                    ; 15/04/2019 - Retro DOS v4.0
                                
                                    ;--------------------------------------------------------------------------
                                    ;
                                    ; procedure : parseline
                                    ;
                                    ; entry point is parseline. al contains the first character in command line.
                                    ;
                                    ;--------------------------------------------------------------------------
                                
                                        ; 16/04/2024 - Retrodos v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
                                        ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:4151h)
                                
                                        ; 03/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
                                        ; (SYSINIT:3EDFh)
                                
                                        ; 01/11/2022 - Retrodos v4.0 (Modified MSDOS 5.0 IO.SYS)
                                        ; (SYSINIT:30ACh)
                                
                                    parseline:
                                        ; 03/01/2023
                                        ; ds = cs ; *
                                
                                        ;push  ds ; *
                                
                                        ;push  cs ; *
                                        ;pop   ds ; *
                                
                                    nextswtch:
00003B05 3C0D                           cmp     al,cr                   ; carriage return?
00003B07 741C                           je      short done_line
00003B09 3C0A                           cmp     al,lf                   ; linefeed?
00003B0B 7421                           je      short put_back          ; put it back and done
                                
                                    ; anything less or equal to a space is ignored.
                                
00003B0D 3C20                           cmp     al,' '                    ; space?
00003B0F 760F                           jbe     short getnext           ; skip over space
00003B11 3C2F                           cmp     al,'/'
00003B13 7402                           je      short getparm
00003B15 F9                             stc                             ; mark error invalid-character-in-input
                                        ;jmp   short exitpl
                                        ; 03/01/2023
                                    swterr:
00003B16 C3                             retn
                                
                                    getparm:
00003B17 E86EFF                         call    check_switch
00003B1A 891E[1D4F]                     mov     [switches],bx           ; save switches read so far
00003B1E 72F6                           jc      short swterr
                                    getnext:
00003B20 E88C0C                         call    getchr
                                        ;jc    short done_line
                                        ;jmp   short nextswtch
                                        ; 03/01/2023
00003B23 73E0                           jnc     short nextswtch
                                    ;swterr:
                                        ;jmp   short exitpl            ; exit if error
                                
                                    done_line:
                                        ; 12/12/2022
00003B25 F606[1D4F]08                   test    byte [switches],flagdrive ; 8
                                        ;test  word [switches],flagdrive ; 8 ; see if drive specified
00003B2A 750C                           jnz     short okay
00003B2C F9                             stc                             ; mark error no-drive-specified
                                        ;jmp   short exitpl
                                        ; 03/01/2023
00003B2D C3                             retn
                                
                                    ;exitpl:
                                        ; 03/01/2023
                                        ; ds = cs
                                        ;;pop  ds ; *
                                        ;retn
                                
                                    put_back:
00003B2E FF06[5603]                     inc     word [count]            ; one more char to scan
00003B32 FF0E[5A03]                     dec     word [chrptr]           ; back up over linefeed
00003B36 EBED                           jmp     short done_line
                                
                                    okay:
00003B38 A1[1D4F]                       mov     ax,[switches]
00003B3B 83E003                         and     ax,0003h            ; get flag bits for changeline and non-rem
00003B3E A3[C04D]                       mov     [deviceparameters+A_DEVICEPARAMETERS.DP_DEVICEATTRIBUTES],ax
                                        ; 16/04/2024
                                        ;mov   word [deviceparameters+A_DEVICEPARAMETERS.DP_TRACKTABLEENTRIES],0
                                        ;;;
00003B41 C706[1A4E]0000                 mov     word [deviceparameters+92],0 ; PCDOS 7.1 IBMBIO.COM
                                        ;;;
                                        ;clc                            ; everything is fine
                                        ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                        ; 12/12/2022
                                        ; cf=0
                                        ;clc
                                        ;call  setdeviceparameters
                                        ; 03/01/2023
                                        ;jmp   short setdeviceparameters
                                
                                    %endif
                                
                                    ;   M047 -- Begin modifications (too numerous to mark specifically)
                                
                                    ;--------------------------------------------------------------------------
                                    ;
                                    ; procedure : setdeviceparameters
                                    ;
                                    ; setdeviceparameters sets up the recommended bpb in each bds in the
```

```
                                ; system based on the form factor. it is assumed that the bpbs for the
                                ; various form factors are present in the bpbtable. for hard files,
                                ; the recommended bpb is the same as the bpb on the drive.
                                ; no attempt is made to preserve registers since we are going to jump to
                                ; sysinit straight after this routine.
                                ;
                                ;   if we return carry, the DRIVPARM will be aborted, but presently
                                ;     we always return no carry
                                ;
                                ;   note:  there is a routine by the same name in msdioctl.asm
                                ;
                                ;--------------------------------------------------------------------------

                                ; 15/04/2019 - Retro DOS v4.0

                                    ; 02/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)

                                    ; 03/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:3FC4h)

                                    ; 09/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:4236h)

                                setdeviceparameters:
                                    ; 03/01/2023
                                    ; ds = cs

00003B47 06                         push    es

00003B48 0E                         push    cs
00003B49 07                         pop     es

00003B4A 31DB                       xor     bx,bx
00003B4C 8A1E[BF4D]                 mov     bl,[deviceparameters+A_DEVICEPARAMETERS.DP_DEVICETYPE]
00003B50 80FB00                     cmp     bl,DEV_5INCH ; 0
00003B53 7506                       jne     short got_80

00003B55 C706[C24D]2800             mov     word [deviceparameters+A_DEVICEPARAMETERS.DP_CYLINDERS],40
                                                                            ; 48 tpi=40 cyl
                                got_80:
00003B5B D1E3                       shl     bx,1                ; get index into bpb table
00003B5D 8BB7[2E50]                 mov     si,[bpbtable+bx]    ; get address of bpb

                                    ;mov    di,deviceparameters+7
                                    ; 02/11/2022
00003B61 BF[C54D]                   mov     di,deviceparameters+A_DEVICEPARAMETERS.DP_BPB ; es:di -> bpb
00003B64 B93B00                     mov     cx,A_BPB.size ; 31
                                    ; 09/09/2023
                                    ;mov    cx,59 ; PCDOS 7.1 IBMBIO.COM A_BPB.size
00003B67 FC                         cld
                                    ;repe   movsb
                                    ; 02/11/2022
00003B68 F3A4                       rep     movsb

00003B6A 07                         pop     es

                                    ; 12/12/2022
00003B6B F606[1D4F]20               test    byte [switches],flagseclim ; 20h
                                    ;test   word [switches],flagseclim ; 20h
00003B70 7406                       jz      short see_heads

00003B72 A1[1A4F]                   mov     ax,[slim]
                                    ;mov    [deviceparameters+20],ax
00003B75 A3[D24D]                   mov     [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERTRACK],ax

                                see_heads:
                                    ; 12/12/2022
00003B78 F606[1D4F]40               test    byte [switches],flagheads ; 40h
                                    ;test   word [switches],flagheads ; 40h
00003B7D 7406                       jz      short heads_not_altered

00003B7F A1[184F]                   mov     ax,[hlim]
                                    ;mov    [deviceparameters+22],ax
00003B82 A3[D44D]                   mov     [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.HEADS],ax

                                heads_not_altered:

                                ; set up correct media descriptor byte and sectors/cluster
                                ;   sectors/cluster is always 2 except for any one sided disk or 1.44M

                                    ;mov    byte [deviceparameters+9],2
                                    ; 02/11/2022
                                    ;mov    byte [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERCLUSTER],2
                                    ; 03/01/2023
00003B85 B80200                     mov     ax,2
00003B88 A2[C74D]                   mov     [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERCLUSTER],al ; 2

00003B8B B3F0                       mov     bl,0F0h             ; get default mediabyte

                                ;   preload the mediadescriptor from the bpb into bh for convenient access

                                    ;mov    bh,[deviceparameters+17]
                                    ; 02/11/2022
00003B8D 8A3E[CF4D]                 mov     bh,[deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.MEDIADESCRIPTOR]

                                    ; 03/01/2023
                                    ; ax = 2
00003B91 3906[D44D]                 cmp     [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.HEADS],ax ; >2 heads?
                                    ;cmp    word [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.HEADS],2 ; >2 heads?
00003B95 773C                       ja      short got_correct_mediad ; just use default if heads>2

00003B97 7524                       jne     short only_one_head   ; one head, do one head stuff

                                ;   two head drives will use the mediadescriptor from the bpb

00003B99 88FB                       mov     bl,bh               ; get mediadescriptor from bpb

                                ;   two sided drives have two special cases to look for. One is
                                ;      a 320K diskette (40 tracks, 8 secs per track). It uses
                                ;      a mediaid of 0fch. The other is 1.44M, which uses only
                                ;      one sector/cluster.

                                ;   any drive with 18secs/trk, 2 heads, 80 tracks, will be assumed
                                ;      to be a 1.44M and use only 1 sector per cluster. Any other
                                ;      type of 2 headed drive is all set.

00003B9B 833E[D24D]12               cmp     word [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERTRACK],18
00003BA0 7509                       jne     short not_144m
00003BA2 833E[C24D]50               cmp     word [deviceparameters+A_DEVICEPARAMETERS.DP_CYLINDERS],80
00003BA7 7502                       jne     short not_144m

                                ;   We've got cyl=80, heads=2, secpertrack=18. Set cluster size to 1.

00003BA9 EB24                       jmp     short got_one_secperclus_drive
```

```
42001
42002                                   ;   check for 320K
42003
42004                                   not_144m:
42005 00003BAB 833E[C24D]28                 cmp     word [deviceparameters+A_DEVICEPARAMETERS.DP_CYLINDERS],40
42006 00003BB0 7521                         jne     short got_correct_mediad
42007 00003BB2 833E[D24D]08                 cmp     word [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERTRACK],8
42008 00003BB7 751A                         jne     short got_correct_mediad
42009
42010 00003BB9 B3FC                         mov     bl,0FCh
42011 00003BBB EB16                         jmp     short got_correct_mediad
42012
42013                                   only_one_head:
42014
42015                                   ;   if we don't have a 360K drive, then just go use 0f0h as media descr.
42016
42017 00003BBD 803E[BF4D]00                 cmp     byte [deviceparameters+A_DEVICEPARAMETERS.DP_DEVICETYPE],DEV_5INCH ; 0
42018 00003BC2 740B                         je      short got_one_secperclus_drive
42019
42020                                   ;   single sided 360K drive uses either 0fch or 0feh, depending on
42021                                   ;     whether sectorspertrack is 8 or 9. For our purposes, anything
42022                                   ;     besides 8 will be considered 0fch
42023
42024 00003BC4 B3FC                         mov     bl,0FCh                 ; single sided 9 sector media id
42025 00003BC6 833E[D24D]08                 cmp     word [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERTRACK],8
42026                                       ; 12/12/2022
42027 00003BCB 7502                         jne     short got_one_secperclus_drive ; okay if anything besides 8
42028
42029 00003BCD B3FE                         mov     bl,0FEh                 ; 160K mediaid
42030
42031                                   ;   we've either got a one sided drive, or a 1.44M drive
42032                                   ;     either case we'll use 1 sector per cluster instead of 2
42033
42034                                   got_one_secperclus_drive:
42035                                       ; 03/01/2023
42036                                       ; ax = 2
42037 00003BCF 48                           dec     ax  ; ax = 1
42038 00003BD0 A2[C74D]                     mov     [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERCLUSTER],al ; 1
42039                                       ;mov    byte [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERCLUSTER],1
42040
42041                                   got_correct_mediad:
42042 00003BD3 881E[CF4D]                   mov     [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.MEDIADESCRIPTOR],bl
42043
42044                                   ;    Calculate the correct number of Total Sectors on medium
42045
42046 00003BD7 A1[C24D]                     mov     ax,[deviceparameters+A_DEVICEPARAMETERS.DP_CYLINDERS]
42047 00003BDA F726[D44D]                   mul     word [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.HEADS]
42048 00003BDE F726[D24D]                   mul     word [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERTRACK]
42049 00003BE2 A3[CD4D]                     mov     word [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.TOTALSECTORS],ax
42050 00003BE5 F8                           clc                             ; we currently return no errors
42051
42052 00003BE6 C3                           retn
42053
42054                                   ;   M047 -- end rewritten routine
42055
42056                                   ;--------------------------------------------------------------------------
42057                                   ;
42058                                   ; procedure : organize
42059                                   ;
42060                                   ;--------------------------------------------------------------------------
42061
42062                                   ; 09/09/2023 - Retrodos v4.2 IO.SYS (Optimization)
42063                                   %if 1
42064                                   end_commd_line:
42065 00003BE7 AA                           stosb                           ; store line feed char in buffer for the linecount.
42066                                       ;mov    byte [cs:com_level],0 ; reset the command level.
42067                                       ; 03/01/2023
42068                                       ; ds = cs
42069                                       ;mov    byte [com_level],0
42070                                       ;jmp    short org1
42071                                       ; 09/09/2023
42072 00003BE8 EB0E                         jmp     short org0
42073                                   nochar1:
42074 00003BEA F9                           stc
42075 00003BEB C3                           retn
42076                                   %endif
42077                                       ; 02/11/2022 - Retrodos v4.0 (Modified MSDOS 5.0 IO.SYS)
42078                                       ; (SYSINIT:3234h)
42079
42080                                       ; 03/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
42081                                       ; (SYSINIT:4067h)
42082
42083                                       ; 09/09/2023 - Retrodos v4.2 IO.SYS (Optimization)
42084                                       ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:42D9h)
42085
42086                                   organize:
42087                                       ; 03/01/2023
42088                                       ; ds = cs
42089 00003BEC 8B0E[5603]                   mov     cx,[count]
42090                                       ;mov    cx,[cs:count]
42091 00003BF0 E3F8                         jcxz    nochar1
42092
42093                                   ;ifndef    MULTI_CONFIG
42094                                   ;
42095                                   ;;   In MULTI_CONFIG, we map to upper case on a line-by-line basis,
42096                                   ;;   because we the case of values in SET commands preserved
42097                                   ;
42098                                   ;   call    mapcase
42099                                   ;endif
42100                                       ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
42101                                       ; 03/01/2023 - Retro DOS v4.2
42102                                       ;call   mapcase
42103
42104 00003BF2 31F6                         xor     si,si
42105 00003BF4 89F7                         mov     di,si
42106 00003BF6 31C0                         xor     ax,ax
42107                                       ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
42108                                       ;;mov   byte [cs:com_level],0
42109                                       ; 12/12/2022
42110                                       ;mov    [cs:com_level],al ; 0
42111                                       ; 03/01/2023
42112                                       ; ds = cs
42113                                       ; 09/09/2023
42114                                       ;mov    [com_level],al ; 0
42115                                   org0:
42116 00003BF8 C606[5003]00                 mov     byte [com_level],0 ; 09/09/2023
42117                                   org1:
42118 00003BFD E8EF01                       call    skip_comment
42119 00003C00 74E5                         jz      short end_commd_line  ; found a comment string and skipped.
42120 00003C02 E8D001                       call    get2                  ; not a comment string. then get a char.
42121 00003C05 3C0A                         cmp     al,lf ; 0Ah
42122 00003C07 74DE                         je      short end_commd_line  ; starts with a blank line.
42123 00003C09 3C20                         cmp     al,' ' ; 20h
42124 00003C0B 76F0                         jbe     short org1            ; skip leading control characters
```

```
                                    ; 09/09/2023
                                    ;jmp    short findit

                                ; 09/09/2023
                                %if 0
                                end_commd_line:
                                    stosb                           ; store line feed char in buffer for the linecount.
                                    ;mov    byte [cs:com_level],0 ; reset the command level.
                                    ; 03/01/2023
                                    ; ds = cs
                                    mov     byte [com_level],0
                                    jmp     short org1

                                nochar1:
                                    stc
                                    retn
                                %endif

                                findit:
00003C0D 51                         push    cx
00003C0E 56                         push    si
00003C0F 57                         push    di
00003C10 89F5                       mov     bp,si
00003C12 4D                         dec     bp
00003C13 BE[D24C]                       mov     si,comtab           ; prepare to search command table
00003C16 B500                       mov     ch,0
                                findcom:
00003C18 89EF                       mov     di,bp
00003C1A 8A0C                       mov     cl,[si]
00003C1C 46                         inc     si
00003C1D E345                       jcxz    nocom

                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)

                                ; 03/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)

                                ;ifdef      MULTI_CONFIG
                                ;   Simplify future parsing by collapsing ";" onto "REM", and at the same
                                ;   time skip the upcoming delimiter test (since ";" need not be followed by
                                ;   anything in particular)

00003C1F 26803D3B                   cmp     byte [es:di],CONFIG_SEMICOLON  ; ';'
00003C23 7430                       je      short semicolon
                                loopcom:
                                    ;mov    al,[es:di]
                                    ;inc    di
                                    ;and    al,~20h ; 0DFh          ; force upper case
                                    ;inc    si                        ; compare to byte @es:di
                                    ;cmp    al,[si-1]
                                    ; 28/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
00003C25 268A25                     mov     ah,[es:di]
00003C28 47                         inc     di
00003C29 80E4DF                     and     ah,~20h ; 0DFh
00003C2C AC                         lodsb                   ; mov al,[si]
                                                            ; inc si
                                    ;cmp    al,ah
                                    ;loope  loopcom
                                    ; 28/07/2023
00003C2D 30C4                       xor     ah,al           ; result: ah = 0 (*) if ah = al
00003C2F E1F4                       loopz   loopcom
                                ;else
                                ;   repe    cmpsb
                                ;endif
                                    ; 02/11/2022
                                    ; 03/01/2023 - Retro DOS v4.2
                                    ;repe   cmpsb

                                    ; 28/07/2023
                                    ;lahf
00003C31 01CE                           add     si,cx                   ; bump to next position without affecting flags
                                    ;sahf
00003C33 AC                             lodsb                           ; get indicator letter
                                    ;jnz    short findcom
                                        ; 28/07/2023
00003C34 08E4                       or      ah,ah                   ; (*)
00003C36 75E0                       jnz     short findcom

00003C38 26803D0D                   cmp     byte [es:di],cr             ; the next char might be cr,lf
00003C3C 7421                       je      short gotcom0           ; such as in "rem",cr,lf case.
00003C3E 26803D0A                   cmp     byte [es:di],lf
00003C42 741B                       je      short gotcom0

                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)

                                ; 03/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)

                                ;ifdef      MULTI_CONFIG

                                ; Skip the delimiter test for the BEGIN identifier (it doesn't have one).

00003C44 3C5B                       cmp     al,CONFIG_BEGIN  ; '['
00003C46 7417                       je      short gotcom0
                                ;endif
00003C48 50                         push    ax
00003C49 268A05                     mov     al,[es:di]              ; now the next char. should be a delim.

                                ;ifdef      MULTI_CONFIG

                                ;   If keyword is *immediately* followed by a question mark (?), then
                                ;   set the high bit of the ASCII command code (CONFIG_OPTION_QUERY) that is
                                ;   stored in the CONFIG.SYS memory image.

00003C4C 3C3F                       cmp     al,'?'                    ; explicit interactive command?
00003C4E 7509                       jne     short no_query          ; no
00003C50 58                         pop     ax                        ; yes, so retrieve the original code
                                    ;or     al,80h ; 03/01/2023
00003C51 0C80                       or      al,CONFIG_OPTION_QUERY  ; and set the QUERY bit
00003C53 EB0A                       jmp     short gotcom0           ;
                                semicolon:
00003C55 B030                       mov     al,CONFIG_REM ; '0'
00003C57 EB06                       jmp     short gotcom0
                                no_query:
                                ;endif  ;MULTI_CONFIG

                                    ; 02/11/2022
                                    ; 03/01/2023 - Retro DOS v4.2
                                    ;push   ax
                                    ;mov    al,[es:di]              ; now the next char. should be a delim.

00003C59 E82E0B                     call    delim
                                no_delim:
00003C5C 58                         pop     ax
00003C5D 75B9                       jnz     short findcom
```

```
                                gotcom0:
00003C5F 5F                         pop     di
00003C60 5E                         pop     si
00003C61 59                         pop     cx
00003C62 EB10                       jmp     short gotcom
                                nocom:
00003C64 5F                         pop     di
00003C65 5E                         pop     si
00003C66 59                         pop     cx
00003C67 B05A                           mov     al,CONFIG_UNKNOWN  ; 'Z'
00003C69 AA                         stosb                           ; save indicator char.
                                _skipline:
00003C6A E86801                     call    get2
00003C6D 3C0A                       cmp     al,lf ; 0Ah             ; skip this bad command line
00003C6F 75F9                           jne     short _skipline
                                    ;jmp    short end_commd_line    ; handle next command line
                                    ; 09/09/2023
00003C71 E973FF                     jmp     end_commd_line
                                gotcom:
00003C74 AA                             stosb                               ; save indicator char in buffer

                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)

                                ; 03/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)

                                ;ifdef      MULTI_CONFIG

                                ;   Don't pollute "cmd_indicator" with the CONFIG_OPTION_QUERY bit though;
                                ;   it screws up the direct comparisons below.

00003C75 247F                       and     al,~CONFIG_OPTION_QUERY ; 7Fh
                                ;endif
                                    ;mov    [cs:cmd_indicator],al ; save it for the future use.
                                    ; 03/01/2023
                                    ; ds = cs
00003C77 A2[5403]                   mov     [cmd_indicator],al    ; save it for the future use.

                                ;ifdef      MULTI_CONFIG

                                ;   There is no whitespace/delimiter between the "begin block" character
                                ;   ([) and the name of block (eg, [menu]), therefore skip this delimiter
                                ;   skipping code

00003C7A 3C5B                       cmp     al,CONFIG_BEGIN
00003C7C 7455                       je      short org31
00003C7E 3C4F                       cmp     al,CONFIG_SUBMENU ; 'O'
00003C80 740F                       je      short no_mapcase
00003C82 3C45                       cmp     al,CONFIG_MENUITEM ; 'E'
00003C84 740B                       je      short no_mapcase
00003C86 3C41                       cmp     al,CONFIG_MENUDEFAULT ; 'A'
00003C88 7407                       je      short no_mapcase
00003C8A 3C4A                       cmp     al,CONFIG_INCLUDE ; 'J'
00003C8C 7403                       je      short no_mapcase
00003C8E E8350B                     call    mapcase                 ; map case of rest of line to UPPER
                                no_mapcase:
                                ;endif
                                    ;; 02/11/2022
                                    ;;mov   [cs:cmd_indicator],al ; save it for the future use.
                                    ;; 03/01/2023
                                    ;; ds = cs
                                    ;mov    [cmd_indicator],al
                                org2:
00003C91 E84101                     call    get2                    ; skip the command name until delimiter
00003C94 3C0A                           cmp     al,lf           ; 0Ah
00003C96 740F                       je      short org21
00003C98 3C0D                       cmp     al,cr       ; 0Dh
00003C9A 740B                       je      short org21
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ; 03/01/2023 - Retro DOS v4.2
00003C9C 3C2F                       cmp     al, '/'                 ; T-RICHJ: Added to allow DEVHIGH/L:...
00003C9E 7407                       je      short org21             ; T-RICHJ: to be parsed properly.

00003CA0 E8E70A                     call    delim
00003CA3 75EC                           jnz short org2
00003CA5 EB02                       jmp     short org3
                                org21:                                  ;if cr or lf then
00003CA7 4E                         dec     si                      ; undo si, cx register
00003CA8 41                         inc     cx                      ;  and continue
                                org3:
                                    ;cmp    byte [cs:cmd_indicator],CONFIG_COMMENT ; 'Y'
                                    ;je     short get_cmt_token
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ; 03/01/2023 - Retro DOS v4.2
                                    ;cmp    byte [cs:cmd_indicator],CONFIG_DEVICE ; 'D'
                                    ;je     short org_file
                                    ;cmp    byte [cs:cmd_indicator],CONFIG_INSTALL ; 'I'
                                    ;je     short org_file
                                    ;cmp    byte [cs:cmd_indicator],CONFIG_INSTALLHIGH ; 'W'
                                    ;je     short org_file
                                    ; 02/11/2022
                                    ; 03/01/2023 - Retro DOS v4.2
                                    ;;cmp   byte [cs:cmd_indicator],CONFIG_DEVICE ; 'D'
                                    ;;je    short org_file
                                    ;cmp    byte [cs:cmd_indicator],CONFIG_SHELL ; 'S'
                                    ;je     short org_file
                                        ;cmp        byte [cs:cmd_indicator],CONFIG_SWITCHES ; '1'
                                    ;je     short org_switch

                                    ; 03/01/2023
                                    ; ds = cs

00003CA9 803E[5403]59               cmp     byte [cmd_indicator],CONFIG_COMMENT ; 'Y'
00003CAE 745D                       je      short get_cmt_token
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ; 03/01/2023 - Retro DOS v4.2
00003CB0 803E[5403]44               cmp     byte [cmd_indicator],CONFIG_DEVICE ; 'D'
00003CB5 7430                       je      short org_file
00003CB7 803E[5403]49               cmp     byte [cmd_indicator],CONFIG_INSTALL ; 'I'
00003CBC 7429                       je      short org_file
00003CBE 803E[5403]57               cmp     byte [cmd_indicator],CONFIG_INSTALLHIGH ; 'W'
00003CC3 7422                       je      short org_file
                                    ; 02/11/2022
                                    ; 03/01/2023 - Retro DOS v4.2
                                    ;cmp    byte [cmd_indicator],CONFIG_DEVICE ; 'D'
                                    ;je     short org_file
00003CC5 803E[5403]53                   cmp byte [cmd_indicator],CONFIG_SHELL ; 'S'
00003CCA 741B                       je      short org_file
00003CCC 803E[5403]31                   cmp byte [cmd_indicator],CONFIG_SWITCHES ; '1'
00003CD1 7403                       je      short org_switch

                                org31:
00003CD3 E99500                     jmp     org4

                                org_switch:
```

```
42373 00003CD6 E81601                   call    skip_comment
42374 00003CD9 7472                     jz      short end_commd_line_brdg
42375
42376 00003CDB E8F700                   call    get2
42377 00003CDE E8B10A                   call    org_delim
42378 00003CE1 74F3                     jz      short org_switch
42379
42380 00003CE3 AA                       stosb
42381 00003CE4 E99300                   jmp     org5
42382
42383                                 org_file:                 ; get the filename and put 0 at end
42384 00003CE7 E80501                   call    skip_comment
42385 00003CEA 7464                     jz      short org_put_zero
42386
42387 00003CEC E8E600                   call    get2            ; not a comment
42388 00003CEF E8980A                   call    delim
42389 00003CF2 74F3                     jz      short org_file ; skip the possible delimiters
42390
42391 00003CF4 AA                       stosb                   ; copy the first non delim char found in buffer
42392
42393                                 org_copy_file:
42394 00003CF5 E8F700                   call    skip_comment   ; comment char in the filename?
42395 00003CF8 7456                     jz      short org_put_zero ; then stop copying filename at that point
42396
42397 00003CFA E8D800                   call    get2
42398 00003CFD 3C2F                     cmp     al,'/'          ; a switch char? (device=filename/xxx)
42399 00003CFF 7457                     je      short end_file_slash ; this will be the special case.
42400
42401 00003D01 AA                       stosb                   ; save the char. in buffer
42402 00003D02 E8850A                   call    delim
42403 00003D05 7459                     jz      short end_copy_file
42404
42405 00003D07 3C20                     cmp     al, ' '
42406 00003D09 77EA                     ja      short org_copy_file ; keep copying
42407 00003D0B EB53                     jmp     short end_copy_file ; otherwise, assume end of the filename.
42408
42409                                 get_cmt_token:                  ; get the token. just max. 2 char.
42410 00003D0D E8C500                   call    get2
42411 00003D10 3C20                     cmp     al,' '          ; skip white spaces or "=" char.
42412 00003D12 74F9                     je      short get_cmt_token ; (we are allowing the other special
42413 00003D14 3C09                     cmp     al,tab ; 9      ;  characters can used for comment id.
42414 00003D16 74F5                     je      short get_cmt_token ;  character.)
42415 00003D18 3C3D                     cmp     al,'='          ; = is special in this case.
42416 00003D1A 74F1                     je      short get_cmt_token
42417 00003D1C 3C0D                     cmp     al,cr
42418 00003D1E 7426                     je      short get_cmt_end ; cannot accept the carriage return
42419 00003D20 3C0A                     cmp     al,lf
42420 00003D22 7422                     je      short get_cmt_end
42421
42422                                   ; 03/01/2023
42423                                   ; ds = cs
42424                                   ;mov    [cs:cmmt1],al  ; store it
42425                                   ;mov    byte [cs:cmmt],1 ; 1 char. so far.
42426 00003D24 A2[5203]                 mov     [cmmt1],al      ; store it
42427 00003D27 C606[5103]01             mov     byte [cmmt],1  ; 1 char. so far.
42428 00003D2C E8A600                   call    get2
42429 00003D2F 3C20                     cmp     al,' ' ; 20h
42430 00003D31 7413                     je      short get_cmt_end
42431 00003D33 3C09                     cmp     al,tab ; 9
42432 00003D35 740F                     je      short get_cmt_end
42433 00003D37 3C0D                     cmp     al,cr  ; 0Dh
42434 00003D39 740B                     je      short get_cmt_end
42435 00003D3B 3C0A                     cmp     al,lf  ; 0Ah
42436 00003D3D 740E                     je      short end_commd_line_brdg
42437
42438                                   ;mov    [cs:cmmt2],al
42439                                   ;inc    byte [cs:cmmt]
42440                                   ; 03/01/2023
42441 00003D3F A2[5303]                 mov     [cmmt2],al
42442 00003D42 FE06[5103]               inc     byte [cmmt]
42443
42444                                 get_cmt_end:
42445 00003D46 E88C00                   call    get2
42446 00003D49 3C0A                     cmp     al,lf
42447 00003D4B 75F9                     jne     short get_cmt_end     ; skip it.
42448                                 end_commd_line_brdg:
42449 00003D4D E997FE                   jmp     end_commd_line        ; else jmp to end_commd_line
42450
42451                                 org_put_zero:                           ; make the filename in front of
42452 00003D50 26C60500                 mov     byte [es:di],0        ;  the comment string to be an asciiz.
42453 00003D54 47                       inc     di
42454 00003D55 E98FFE                   jmp     end_commd_line        ;  (maybe null if device=/*)
42455
42456                                 end_file_slash:                 ; al = "/" option char.
42457 00003D58 26C60500                 mov     byte [es:di],0        ; make a filename an asciiz
42458 00003D5C 47                       inc     di                    ; and
42459 00003D5D AA                       stosb                         ; store "/" after that.
42460 00003D5E EB1A                     jmp     short org5            ; continue with the rest of the line
42461
42462                                 end_copy_file:
42463 00003D60 26C645FF00               mov     byte [es:di-1],0      ; make it an asciiz and handle the next char.
42464 00003D65 3C0A                     cmp     al,lf
42465 00003D67 74E4                     je      short end_commd_line_brdg
42466 00003D69 EB0F                     jmp     short org5
42467
42468                                 org4:                                   ; org4 skips all delimiters after the command name except for '/'
42469 00003D6B E88100                   call    skip_comment
42470 00003D6E 74DD                     jz      short end_commd_line_brdg
42471
42472 00003D70 E86200                   call    get2
42473 00003D73 E81C0A                   call    org_delim             ; skip delimiters except '/' (mrw 4/88)
42474 00003D76 74F3                     jz      short org4
42475 00003D78 EB08                     jmp     short org51
42476
42477                                 org5:                                   ; rest of the line
42478 00003D7A E87200                   call    skip_comment          ; comment?
42479 00003D7D 74CE                     jz      short end_commd_line_brdg
42480 00003D7F E85300                   call    get2                  ; not a comment.
42481
42482                                 org51:
42483 00003D82 AA                       stosb                         ; copy the character
42484 00003D83 3C22                     cmp     al,'"'  ; 22h         ; a quote ?
42485 00003D85 743A                     je      short at_quote
42486 00003D87 3C20                     cmp     al,' '  ; 20h
42487 00003D89 77EF                     ja      short org5
42488
42489                                   ; 09/09/2023
42490                                   ; (Note: PCDOS 7.1 IBMBIO.COM does not contain M051 modification)
42491
42492                                                                   ; M051 - Start
42493                                   ; 03/01/2023
42494                                   ; ds = cs
42495 00003D8B 803E[5403]55                 cmp byte [cmd_indicator],CONFIG_DEVICEHIGH
42496                                   ;cmp    byte [cs:cmd_indicator],CONFIG_DEVICEHIGH ; Q: is this devicehigh
```

```
00003D90 7514                       jne     short not_dh            ; N:
00003D92 3C0A                       cmp     al,lf                   ; Q: is this line feed
00003D94 7416                       je      short org_dhlf          ; Y: stuff a blank before the lf
00003D96 3C0D                       cmp     al,cr                   ; Q: is this a cr
00003D98 75E0                       jne     short org5              ; N:
00003D9A 26C645FF20                 mov     byte [es:di-1],' '      ; overwrite cr with blank
00003D9F AA                         stosb                           ; put cr after blank
00003DA0 FE06[223A]                 inc     byte [insert_blank]
                                    ;inc    byte [cs:insert_blank]; indicate that blank has been
                                                                    ; inserted
00003DA4 EBD4                       jmp     short org5
                                not_dh:                                 ; M051 - End

00003DA6 3C0A                       cmp     al,lf                   ; line feed?
00003DA8 740F                       je      short org1_brdg             ; handles the next command line.
00003DAA EBCE                       jmp     short org5              ; handles next char in this line.

                                org_dhlf:                               ; M051 - Start
                                    ; 03/01/2023
                                    ; ds = cs
00003DAC 803E[223A]01               cmp     byte [insert_blank],1
                                    ;cmp    byte [cs:insert_blank],1 ; Q:has a blank already been inserted
00003DB1 7406                       je      short org1_brdg             ; Y:
00003DB3 26C645FF20                 mov     byte [es:di-1],' '      ; overwrite lf with blank
00003DB8 AA                         stosb                           ; put lf after blank
                                                                    ; M051 - End
                                org1_brdg:
00003DB9 C606[223A]00               mov     byte [insert_blank],0
                                    ;mov    byte [cs:insert_blank],0 ; M051: clear blank indicator for
                                                                    ; M051: devicehigh
00003DBE E93CFE                     jmp     org1

                                at_quote:
00003DC1 803E[5003]00               cmp     byte [com_level],0
                                    ;cmp    byte [cs:com_level],0
00003DC6 7407                       je      short up_level
                                    ;mov    byte [cs:com_level],0 ; reset it.
00003DC8 C606[5003]00               mov     byte [com_level],0
00003DCD EBAB                       jmp     short org5

                                up_level:
                                    ;inc    byte [cs:com_level]   ; set it.
00003DCF FE06[5003]                 inc     byte [com_level]
00003DD3 EBA5                       jmp     short org5

                                ;---------------------------------------------------------------------------
                                ;
                                ; procedure : get2
                                ;
                                ;---------------------------------------------------------------------------

                                    ; 02/11/2022 - Retrodos v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:33FAh)

                                    ; 03/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:4270h)
                                get2:
00003DD5 E304                       jcxz    noget
                                    ;
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;;lods  byte ptr es:[si]
                                    ; 12/12/2022
00003DD7 26                         es
00003DD8 AC                         lodsb
                                    ;mov    al, [es:si]
                                    ;inc    si
                                    ;
00003DD9 49                         dec     cx
00003DDA C3                         retn
                                noget:
00003DDB 59                         pop     cx
                                    ; 03/01/2023
                                    ; ds = cs
                                    ;mov    [cs:count],di ; 13/05/2019
                                    ;mov    [cs:org_count],di
00003DDC 893E[5603]                 mov     [count],di
00003DE0 893E[5803]                 mov     [org_count],di
00003DE4 31F6                       xor     si,si
                                    ;mov    [cs:chrptr],si
00003DE6 8936[5A03]                 mov     [chrptr],si

                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)

                                ; 03/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)

                                ;ifndef MULTI_CONFIG
                                ;   retn
                                ;else

                                ;   This was the rather kludgy way out of procedure "organize", but instead
                                ;   of returning to doconf, we now want to check config.sys BEGIN/END blocks
                                ;   and the new boot menu stuff...

00003DEA 89F9                       mov     cx,di
00003DEC E9E300                     jmp     menu_check

                                ;endif
                                    ; 02/11/2022
                                    ; 03/01/2023 - Retro DOS v4.2
                                    ;retn

                                ;---------------------------------------------------------------------------
                                ;
                                ; procedure : skip_comment
                                ;
                                ;skip the commented string until lf, if current es:si-> a comment string.
                                ;in) es:si-> string
                                ;    cx -> length.
                                ;out) zero flag not set if not found a comment string.
                                ;     zero flag set if found a comment string and skipped it. al will contain
                                ;     the line feed character at this moment when return.
                                ;     ax register destroyed.
                                ;     if found, si, cx register adjusted accordingly.
                                ;
                                ;---------------------------------------------------------------------------

                                    ; 03/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:428Dh)

                                skip_comment:
00003DEF E3EA                       jcxz    noget           ; get out of the organize routine.

                                    ; 03/01/2023
                                    ; ds = cs
```

```
42621
42622 00003DF1 803E[5003]00              cmp     byte [com_level],0
42623                                   ;cmp    byte [cs:com_level],0 ; only check it if parameter level is 0.
42624 00003DF6 752C                     jne     short no_commt  ; (not inside quotations)
42625
42626 00003DF8 803E[5103]01             cmp     byte [cmmt],1
42627                                   ;cmp    byte [cs:cmmt],1
42628 00003DFD 7225                     jb      short no_commt
42629
42630 00003DFF 268A04                   mov     al,[es:si]
42631
42632 00003E02 3806[5203]               cmp     [cmmt1],al
42633                                   ;cmp    [cs:cmmt1],al
42634 00003E06 751C                     jne     short no_commt
42635
42636 00003E08 803E[5103]02             cmp     byte [cmmt],2
42637                                   ;cmp    byte [cs:cmmt],2
42638 00003E0D 750A                     jne     short skip_cmmt
42639
42640 00003E0F 268A4401                 mov     al,[es:si+1]
42641
42642 00003E13 3806[5303]               cmp     [cmmt2],al
42643                                   ;cmp    [cs:cmmt2],al
42644 00003E17 750B                     jne     short no_commt
42645                                skip_cmmt:
42646 00003E19 E3C0                     jcxz    noget           ; get out of organize routine.
42647 00003E1B 268A04                   mov     al,[es:si]
42648 00003E1E 46                       inc     si
42649 00003E1F 49                       dec     cx
42650 00003E20 3C0A                     cmp     al,lf           ; line feed?
42651 00003E22 75F5                     jne     short skip_cmmt
42652                                no_commt:
42653 00003E24 C3                       retn
42654
42655                                ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
42656                                ; (SYSINIT:42C8h)
42657
42658                                ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
42659                                ;%if 0
42660
42661                                ;ifdef     MULTI_CONFIG
42662
42663                                ;--------------------------------------------------------------------------
42664                                ;
42665                                ;   kbd_read: wait for keystroke
42666                                ;
42667                                ;   INPUT
42668                                ;       DS == CS == sysinitseg
42669                                ;
42670                                ;   OUTPUT
42671                                ;       Carry SET to clean boot, CLEAR otherwise
42672                                ;
42673                                ;   OTHER REGS USED
42674                                ;       All
42675                                ;
42676                                ;   HISTORY
42677                                ;       Created 16-Nov-1992 by JeffPar
42678                                ;
42679                                ;--------------------------------------------------------------------------
42680
42681                                kbd_read:
42682 00003E25 F606[814C]02                     test    byte [bDisableUI],2
42683 00003E2A 7520                             jnz     short kbd_nodelay
42684
42685 00003E2C 1E                               push    ds              ; the bios timer tick count is incremented
42686 00003E2D 29C0                             sub     ax,ax           ; 18.2 times per second;
42687 00003E2F 8ED8                             mov     ds,ax           ; watch the timer tick count for 37 transitions
42688                                       ;mov   dx,[046Ch]     ; get initial value
42689                                kbd_loop:
42690 00003E31 B401                             mov     ah,1            ;
42691 00003E33 CD16                             int     16h             ; peek the keyboard
42692 00003E35 7514                             jnz short kbd_loopdone ; something's there, get out
42693 00003E37 B402                             mov     ah,2            ; peek the shift states
42694 00003E39 CD16                             int     16h             ;
42695 00003E3B A803                             test    al,03h          ; either right or left shift key bits set?
42696 00003E3D 750C                             jnz short kbd_loopdone ; yes
42697 00003E3F A16C04                           mov     ax,[046Ch] ;
42698                                       ;sub    ax,dx           ; get difference
42699                                       ; 15/04/2019 - Retro DOS v4.0
42700 00003E42 2E2B06[8C03]                   sub     ax,[cs:_timer_lw_] ; MSDOS 6.21 IO.SYS - SYSINIT:42E5h
42701
42702 00003E47 3C25                           cmp     al,37           ; reached limit?  ; (2 seconds)
42703 00003E49 72E6                             jb  short kbd_loop ; not yet
42704                                kbd_loopdone:
42705 00003E4B 1F                               pop     ds              ; delay complete!
42706                                kbd_nodelay:
42707 00003E4C 29DB                             sub     bx,bx           ; assume clean boot
42708 00003E4E B402                             mov     ah,2            ; peek the shift states
42709 00003E50 CD16                             int     16h             ;
42710 00003E52 A803                             test    al,03h          ; either right or left shift key bits set?
42711 00003E54 7407                             jz      short kbd_notshift ; no
42712 00003E56 43                               inc     bx              ; yes
42713 00003E57 43                               inc     bx
42714                                       ; MSDOS 6.21 IO.SYS - SYSINIT:4301h
42715 00003E58 800E[854C]04                   or      byte [bQueryOpt],4
42716                                kbd_notshift:                   ;
42717 00003E5D B401                             mov     ah,1            ; peek the keyboard
42718 00003E5F CD16                             int     16h             ;
42719 00003E61 743E                             jz  short kbd_test ; no key present
42720 00003E63 08C0                             or      al,al           ; is it a function key?
42721 00003E65 753A                             jnz short kbd_test ; no
42722
42723                                       ; MSDOS 6.21 IO.SYS - SYSINIT:430Bh
42724 00003E67 80FC62                           cmp     ah,62h          ; CTRL F5
42725 00003E6A 7405                             je  short kbd_cfg_bypass
42726
42727 00003E6C 80FC3F                           cmp     ah,3Fh          ; F5 function key?
42728 00003E6F 750D                             jne short kbd_notf5       ; no
42729                                kbd_cfg_bypass:
42730 00003E71 BA[FC51]                         mov     dx,_$CleanMsg
42731 00003E74 E8DD0B                           call    print
42732                                       ; MSDOS 6.21 IO.SYS - SYSINIT:431Bh
42733 00003E77 800E[854C]04                   or      byte [bQueryOpt],4
42734 00003E7C EB16                             jmp     short kbd_eat   ; yes, clean boot selected
42735                                kbd_notf5:
42736                                       ; MSDOS 6.21 IO.SYS - SYSINIT:4322h
42737 00003E7E 80FC65                           cmp     ah,65h          ; CTRL F8
42738 00003E81 7405                             je  short kbd_cfg_confirm
42739
42740 00003E83 80FC42                           cmp     ah,42h          ; F8 function key?
42741 00003E86 7523                             jne short kbd_exit ; no
42742                                kbd_cfg_confirm:
42743 00003E88 BA[3A52]                         mov     dx,_$InterMsg
42744 00003E8B E8C60B                           call    print           ;
```

```
42745 00003E8E B301                     mov     bl,1            ; yes, interactive-boot option enabled
42746 00003E90 881E[854C]               mov     [bQueryOpt],bl  ; change default setting
42747                                   kbd_eat:                ;
42748 00003E94 B400                     mov     ah,0            ;
42749 00003E96 CD16                     int     16h             ; eat the key we assumed was a signal
42750 00003E98 C606[8B4C]FF             mov byte [secElapsed],-1
42751 00003E9D 09DB                     or      bx,bx           ;
42752 00003E9F 7405                     jz  short kbd_clean      ;
42753                                   kbd_test:               ;
42754 00003EA1 80FB02                   cmp     bl,2            ;
42755 00003EA4 7205                     jb  short kbd_exit ;
42756                                   kbd_clean:              ;
42757 00003EA6 E86E08                   call    disable_autoexec; yes, tell COMMAND to skip autoexec.bat
42758 00003EA9 F9                       stc                     ; set carry to indicate abort
42759 00003EAA C3                       retn                     ;
42760                                   kbd_exit:               ;
42761 00003EAB F8                       clc                     ; clear carry to indicate success
42762 00003EAC C3                       retn                     ;
42763
42764                                   ;--------------------------------------------------------------------------
42765                                   ;
42766                                   ;   set_numlock: set numlock LED
42767                                   ;
42768                                   ;   INPUT
42769                                   ;       ES:SI -> numlock setting (ie, "ON" or "OFF")
42770                                   ;
42771                                   ;   OUTPUT
42772                                   ;       None
42773                                   ;
42774                                   ;   OTHER REGS USED
42775                                   ;       None
42776                                   ;
42777                                   ;   HISTORY
42778                                   ;       Created 16-Nov-1992 by JeffPar
42779                                   ;
42780                                   ;--------------------------------------------------------------------------
42781
42782                                       ; 04/01/2023 - Retro DOS v4.2
42783
42784                                   set_numlock:
42785                                           ; 04/01/2023
42786                                       ;push   ax
42787 00003EAD 1E                           push    ds
42788 00003EAE 29C0                         sub     ax,ax
42789 00003EB0 8ED8                         mov     ds,ax
42790 00003EB2 268B04                       mov     ax,[es:si]      ; get 1st 2 bytes of value (ON or OF)
42791 00003EB5 2E3B06[C451]                 cmp     ax,[cs:OnOff+2]   ; should we turn it off?
42792 00003EBA 7507                         jne short not_off  ; no
42793 00003EBC 80261704DF                   and     byte [0417h],~20h ; 0DFh
42794 00003EC1 EB0D                         jmp     short set_done
42795                                   not_off:
42796 00003EC3 2E3B06[C251]                 cmp     ax,[cs:OnOff]     ; should we turn it on?
42797 00003EC8 F9                           stc
42798 00003EC9 7505                         jne short set_done ; no
42799 00003ECB 800E170420                   or      byte [0417h],20h
42800                                   set_done:
42801 00003ED0 1F                           pop     ds
42802                                       ; 04/01/2023
42803                                       ;pop    ax
42804 00003ED1 C3                           retn
42805
42806                                   ; 16/04/2019 - Retro DOS v4.0
42807
42808                                   ;--------------------------------------------------------------------------
42809                                   ;
42810                                   ;   menu_check:  check for presence of menu (and other) configuration blocks
42811                                   ;
42812                                   ;   INPUT
42813                                   ;       CX == "organized" config.sys memory image length
42814                                   ;        ES:SI -> "organized" config.sys memory image
42815                                   ;       DS == CS == sysinitseg
42816                                   ;
42817                                   ;   OUTPUT
42818                                   ;       Same as above; the idea is that menu_check simply transforms
42819                                   ;       a block-structured config.sys image into a conventional image,
42820                                   ;       based on the user's block selection and any other boot-time options
42821                                   ;       the user may have employed...
42822                                   ;
42823                                   ;   OTHER REGS USED
42824                                   ;       All
42825                                   ;
42826                                   ;   NOTES
42827                                   ;       [count] and [org_count] are set to the new config.sys image length
42828                                   ;
42829                                   ;   HISTORY
42830                                   ;       Created 16-Mar-1992 by JeffPar
42831                                   ;
42832                                   ;--------------------------------------------------------------------------
42833
42834                                       ; 04/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
42835                                       ; (SYSINIT:4378h)
42836
42837                                   menu_check:
42838
42839                                   ;   Search for SWITCHES, determine if /N or /F are present; if so, then
42840                                   ;   disable clean/interactive boot options
42841
42842 00003ED2 51                           push    cx              ;
42843 00003ED3 56                           push    si              ;
42844 00003ED4 29DB                         sub     bx,bx           ; remains ZERO until first block
42845                                   swchk_loop:                 ;
42846 00003ED6 E83507                       call    get_char        ; get first char of current line
42847 00003ED9 724C                         jc  short swchk_end       ; hit eof
42848 00003EDB 3C5B                         cmp     al,CONFIG_BEGIN ; '['
42849 00003EDD 7503                         jne short swchk_next1 ;
42850 00003EDF 43                           inc     bx              ; remember that we've seen a block
42851 00003EE0 EB40                         jmp     short swchk_nextline
42852                                   swchk_next1:                ;
42853 00003EE2 3C4E                         cmp     al,CONFIG_NUMLOCK
42854 00003EE4 750E                         jne short swchk_next2 ;
42855 00003EE6 09DB                         or      bx,bx           ; only do NUMLOCK commands that exist
42856 00003EE8 7538                         jnz short swchk_nextline ; before the first block
42857 00003EEA E8C0FF                       call    set_numlock     ; REM it out so we don't act on it later, too
42858 00003EED 26C644FF30                   mov     byte [es:si-1],CONFIG_REM
42859 00003EF2 EB2E                         jmp     short swchk_nextline
42860                                   swchk_next2:                ;
42861 00003EF4 3C31                         cmp     al,CONFIG_SWITCHES
42862 00003EF6 752A                         jne short swchk_nextline ; this line ain't it
42863                                   swchk_scan:                 ;
42864 00003EF8 E81307                       call    get_char        ; look for /N or /F
42865                                   swchk_scan1:                ;
42866 00003EFB 3C0A                         cmp     al,LF           ; end of line?
42867 00003EFD 7423                         je  short swchk_nextline ; yes
42868 00003EFF 3C2F                         cmp     al,'/'          ; switch-char?
```

```
00003F01 75F5                           jne short swchk_scan ; no
00003F03 E80807                         call    get_char        ;
00003F06 24DF                           and     al,~20h ; 0DFh    ; convert to upper case
00003F08 3A06[6323]                     cmp     al,[swit_n+1]     ; 'N'
00003F0C 7507                           jne short swchk_scan2 ; no
00003F0E 800E[814C]01                   or      byte [bDisableUI],1
00003F13 EBE3                           jmp short swchk_scan ; continue looking for switches of interest
                                swchk_scan2:                    ;
00003F15 3A06[6F23]                     cmp     al,[swit_f+1]     ; 'F'
00003F19 75E0                           jne short swchk_scan1 ; no
00003F1B 800E[814C]02                   or      byte [bDisableUI],2
00003F20 EBD6                           jmp     short swchk_scan ; continue looking for switches of interest
                                swchk_nextline:                 ;
00003F22 E8C306                         call    skip_opt_line   ;
00003F25 EBAF                           jmp     short swchk_loop ;
                                swchk_end:                      ;
00003F27 5E                             pop     si              ;
00003F28 59                             pop     cx              ;

                                ;   Do the keyboard tests for clean/interactive boot now, but only if
                                ;   the DisableUI flag is still clear

00003F29 F606[814C]01                   test    byte [bDisableUI],1
00003F2E 7508                           jnz short menu_search
                                ;
                                ;   Wait for 2 seconds first, UNLESS the /F bit was set in bDisableUI, or
                                ;   there is anything at all in the keyboard buffer
                                ;
00003F30 E8F2FE                         call    kbd_read
00003F33 7303                           jnc short menu_search
00003F35 E9EE01                         jmp menu_abort

                                ;   Search for MENU block; it is allowed to be anywhere in config.sys

                                menu_search:
00003F38 29DB                           sub     bx,bx           ; if no MENU, default to zero for no_selection
00003F3A BF[C64C]                       mov     di,szMenu ;
00003F3D E80304                         call    find_block      ; find the MENU block
00003F40 7337                           jnc short menu_found ;
00003F42 C606[BE4C]00                   mov     byte [szBoot],0
00003F47 E90C02                         jmp no_selection ; not found

                                ;   Process the requested menu color(s)

                                menu_color:
00003F4A 51                             push    cx              ;
00003F4B 52                             push    dx              ;
                                        ;;mov   dx,0007h        ; default color setting
                                        ; 10/09/2023
                                        ;mov    dl,7 ; !*!
00003F4C E89E06                         call    get_number      ; get first number
00003F4F 80E30F                         and     bl,0Fh  ; !**! ; first # is foreground color (for low nibble)
00003F52 88DD                           mov     ch,bl           ; save it in CH
                                        ; 01/08/2023 - Retro DOS v4.2 IO.SYS (optimization) by Erdogan Tan
                                        ; (high nibble of dl is 0)
                                        ;and    dl,0F0h ; !*!   ; (low nibble of dl would be zero)
                                        ;or     dl,bl           ; (low nibble of dl is 7) ! 14/08/2023
00003F54 88DA                           mov     dl,bl   ; 14/08/2023
00003F56 E83108                         call    delim           ; did we hit a delimiter
00003F59 750E                           jne     short check_color ; no, all done
00003F5B E88F06                         call    get_number      ; get next number
00003F5E 80E30F                         and     bl,0Fh          ; second # is background color (for high nibble)
00003F61 88DE                           mov     dh,bl           ; save it in DH
                                        ; 10/09/2023
                                        ;and    dl,0Fh  ; !**!  ;
00003F63 B104                           mov     cl,4            ;
00003F65 D2E3                           shl     bl,cl           ;
00003F67 08DA                           or      dl,bl           ;
                                check_color:
00003F69 38F5                           cmp     ch,dh           ; are foreground/background the same?
00003F6B 7503                           jne     short set_color       ; no
00003F6D 80F208                         xor     dl,08h          ; yes, so modify the fgnd intensity
                                set_color:
00003F70 8816[7C4C]                     mov     [bMenuColor],dl ;
00003F74 5A                             pop     dx              ;
00003F75 59                             pop     cx              ;
00003F76 E9A900                         jmp     menu_nextitem

                                ;   Back to our regularly scheduled program (the COLOR and other goop
                                ;   above is there simply to alleviate short jump problems)

                                menu_found:
00003F79 C606[864C]01                       mov     byte [bDefBlock],1
                                            ;mov       word [offDefBlock],0
00003F7E C606[8A4C]FF                       mov     byte [secTimeOut],-1
00003F83 8026[854C]FD                       and     byte [bQueryOpt],~2 ; 0FDh
                                        ; 10/09/2023
00003F88 29D2                           sub     dx,dx
00003F8A 8916[884C]                     mov     [offDefBlock],dx ; 0

00003F8E E85706                             call    skip_opt_line   ; skip to next line
                                        ; 10/09/2023
                                        ;sub    dx,dx           ; initialize total block count (0 => none yet)

                                ;   Process the menu block now

                                menu_process:
00003F91 E87A06                         call    get_char        ; get first char of current line
00003F94 722E                           jc  short to_menu_getdefault ; could happen if menu block at end (rare)
00003F96 247F                           and     al,~CONFIG_OPTION_QUERY ; 7Fh
00003F98 3C5B                           cmp     al,CONFIG_BEGIN ; BEGIN implies END
00003F9A 7428                           je  short to_menu_getdefault
00003F9C 3C4F                           cmp     al,CONFIG_SUBMENU
00003F9E 744D                           je  short menu_item       ; go process sub-menu
00003FA0 3C45                           cmp     al,CONFIG_MENUITEM
00003FA2 7449                           je  short menu_item       ; go process menu item
00003FA4 3C41                           cmp     al,CONFIG_MENUDEFAULT
00003FA6 741E                           je  short menu_default ; go process menu default
00003FA8 3C52                           cmp     al,CONFIG_MENUCOLOR
00003FAA 749E                           je  short menu_color ; go process menu color
00003FAC 3C4E                           cmp     al,CONFIG_NUMLOCK
00003FAE 740F                           je  short menu_numlock ;
00003FB0 3C30                           cmp     al,CONFIG_REM   ; allow remarks in menu block
00003FB2 746E                           je  short menu_nextitem ;
00003FB4 E8C307                         call    any_delim       ; allow blank lines and such
00003FB7 7469                           je  short menu_nextitem ;
00003FB9 F9                             stc                     ;
00003FBA E82607                         call    print_error     ; non-MENU command!
00003FBD EB63                           jmp     short menu_nextitem
                                menu_numlock:
00003FBF E8EBFE                         call    set_numlock
00003FC2 EB5E                           jmp     short menu_nextitem
                                to_menu_getdefault:
00003FC4 EB62                           jmp     short menu_getdefault
```

```
42993
42994                                   ;   Save the offset of the default block name, we'll need it later
42995
42996                                   menu_default:
42997 00003FC6 8936[884C]                       mov     [offDefBlock],si; save address of default block name
42998 00003FCA 803E[8B4C]00                     cmp     byte [secElapsed],0
42999 00003FCF 751A                             jne short timeout_skip ; secElapsed is only zero for the FIRST menu,
43000 00003FD1 E8EA05                           call    skip_token      ; and for subsequent menus IF nothing was typed;
43001 00003FD4 724C                             jc  short menu_nextitem ; secElapsed becomes -1 forever as soon as
43002 00003FD6 E8FB05                           call    skip_delim      ; something is typed
43003 00003FD9 7247                             jc  short menu_nextitem ;
43004 00003FDB 89DE                             mov     si,bx           ;
43005 00003FDD E80D06                           call    get_number      ; get number (of seconds for timeout)
43006 00003FE0 80FB5A                           cmp     bl,90           ; limit it to a reasonable number
43007                                       ;jb    short timeout_ok ; (besides, 99 is the largest # my simple
43008 00003FE3 7602                             jna short timeout_ok ; 01/08/2023
43009 00003FE5 B35A                         mov     bl,90           ;  display function can handle)
43010                                   timeout_ok:                     ;
43011 00003FE7 881E[8A4C]                       mov     [secTimeOut],bl ;
43012                                   timeout_skip:
43013 00003FEB EB35                             jmp     short menu_nextitem
43014
43015                                   ;   Verify that this is a valid menu item by searching for the named block
43016
43017                                   menu_item:
43018                                       ;cmp   dl,9    ; 04/01/2023
43019 00003FED 80FA09                           cmp     dl,MAX_MULTI_CONFIG ; have we reached the max # of items yet?
43020 00003FF0 7330                             jae short menu_nextitem ;
43021 00003FF2 89F7                             mov     di,si           ; DS:DI -> block name to search for
43022 00003FF4 E83303                           call    srch_block      ;
43023 00003FF7 7406                             je  short menu_itemfound ;
43024 00003FF9 F9                               stc                     ;
43025 00003FFA E8E606                           call    print_error     ; print error and pause
43026 00003FFD EB23                             jmp     short menu_nextitem ; if not found, ignore this menu item
43027
43028                                   ;   srch_block, having succeeded, returns DI -> past the token that it
43029                                   ;   just matched, which in this case should be a descriptive string; ES:SI
43030                                   ;   and CX are unmodified
43031
43032                                   menu_itemfound:
43033 00003FFF 42                               inc     dx              ; otherwise, increment total block count
43034 00004000 89D3                             mov     bx,dx           ; and use it to index the arrays of offsets
43035 00004002 8887[8C4C]                       mov [abBlockType+bx],al
43036 00004006 01DB                             add     bx,bx           ; of recorded block names and descriptions
43037
43038                                   ;   There should be a description immediately following the block name on
43039                                   ;   MENUITEM line; failing that, we'll just use the block name as the
43040                                   ;   description...
43041
43042 00004008 89B7[964C]                       mov     [aoffBlockName+bx],si
43043 0000400C 89B7[AA4C]                       mov     [aoffBlockDesc+bx],si
43044 00004010 89DF                             mov     di,bx           ; skip_delim modifies BX, so stash it in DI
43045 00004012 E8A905                           call    skip_token      ;
43046 00004015 720B                             jc  short menu_nextitem ; hit eol/eof
43047 00004017 E8BA05                           call    skip_delim      ;
43048 0000401A 7206                             jc  short menu_nextitem ; hit eol/eof
43049 0000401C 87FB                             xchg    bx,di           ;
43050 0000401E 89BF[AA4C]                       mov     [aoffBlockDesc+bx],di
43051
43052                                   menu_nextitem:
43053 00004022 E8C305                           call    skip_opt_line   ;
43054 00004025 E969FF                           jmp     menu_process    ; go back for more lines
43055
43056                                   ;   Display menu items now, after determining which one is default
43057
43058                                   menu_getdefault:
43059 00004028 08D2                             or      dl,dl           ; where there any valid blocks at all?
43060 0000402A 7505                             jnz short menu_valid ; yes
43061 0000402C 29DB                             sub     bx,bx           ; no, so force autoselect of 0
43062 0000402E E9ED00                           jmp     menu_autoselect ; (meaning: process common blocks only)
43063                                   menu_valid:
43064 00004031 29DB                             sub     bx,bx           ;
43065 00004033 8816[874C]                       mov     [bMaxBlock],dl  ; first, record how many blocks we found
43066 00004037 8B3E[884C]                       mov     di,[offDefBlock];
43067 0000403B 09FF                             or      di,di           ; does a default block exist?
43068 0000403D 741C                             jz  short menu_nodefault ; no
43069 0000403F 43                               inc     bx              ; yes, walk name table, looking for default
43070                                   menu_chkdefault:
43071 00004040 53                               push    bx              ;
43072 00004041 01DB                             add     bx,bx           ;
43073 00004043 8BB7[964C]                       mov     si,[aoffBlockName+bx]
43074 00004047 B98000                           mov     cx,128          ; arbitrary maximum length of a name
43075 0000404A 1E                               push    ds              ;
43076 0000404B 06                               push    es              ;
43077 0000404C 1F                               pop     ds              ;
43078 0000404D E81A03                           call    comp_names      ; is this block the same as the default?
43079 00004050 1F                               pop     ds              ;
43080 00004051 5B                               pop     bx              ;
43081 00004052 7409                             je  short menu_setdefault ; yes
43082 00004054 43                               inc     bx              ;
43083 00004055 3A1E[874C]                       cmp     bl,[bMaxBlock]  ; all done searching?
43084 00004059 76E5                             jbe short menu_chkdefault ; not yet
43085                                   menu_nodefault:
43086 0000405B B301                             mov     bl,1            ; if no default, force default to #1
43087                                   menu_setdefault:
43088 0000405D 881E[864C]                       mov     [bDefBlock],bl  ; yes, this will be the initial current block
43089
43090                                   ;   If the timeout was explicitly set to 0 (or technically, anything that
43091                                   ;   failed to resolve to a number, like "NONE" or "EAT POTATOES"), then we're
43092                                   ;   supposed to skip menu display and run with the specified default block;
43093                                   ;   however, if the user hit Enter prior to boot, thereby requesting fully
43094                                   ;   INTERACTIVE boot, then we shall display the menu block anyway (though still
43095                                   ;   with no timeout)
43096
43097 00004061 803E[8A4C]00                       cmp     byte [secTimeOut],0 ; is timeout zero? (ie, assume default)
43098 00004066 750A                             jne short menu_display ; no
43099 00004068 F606[854C]01                     test    byte [bQueryOpt],1 ; yes, but was INTERACTIVE requested?
43100 0000406D 7503                             jnz short menu_display ; yes, so *don't* assume default after all
43101 0000406F E9C700                           jmp     not_topmenu     ;
43102
43103                                   ;   Reset the mode, so that we know screen is clean and cursor is home
43104
43105                                   menu_display:
43106 00004072 B40F                             mov     ah,0Fh          ; get current video mode
43107 00004074 CD10                             int     10h             ;
43108 00004076 B400                             mov     ah,00h          ; just re-select that mode
43109 00004078 CD10                             int     10h             ;
43110 0000407A 06                               push    es              ;
43111 0000407B B84000                           mov     ax,40h          ; reach down into the ROM BIOS data area
43112 0000407E 8EC0                             mov     es,ax           ; and save the current (default) video page
43113 00004080 26A14E00                         mov     ax,[es:004Eh]   ; start address and page #, in case the
43114 00004084 A3[834C]                         mov     [wCRTStart],ax  ; undocumented QUIET option was enabled
43115 00004087 26A06200                         mov     al,[es:0062h]   ;
43116 0000408B A2[824C]                         mov     [bCRTPage],al   ;
```

```
43117 0000408E A1[7D4C]                     mov     ax,[bMenuPage]    ; select new page for menu
43118 00004091 CD10                         int     10h               ;
43119 00004093 B80006                       mov     ax,0600h          ; clear entire screen
43120 00004096 8A3E[7C4C]                   mov     bh,[bMenuColor] ; using this color
43121 0000409A 29C9                         sub     cx,cx             ; upper left row/col
43122                                       ;mov       dl,[es:CRT_Cols]
43123 0000409C 268A164A00                   mov dl,[es:4Ah]
43124 000040A1 FECA                     dec     dl                ;
43125                                       ;mov       dh,[es:CRT_Rows];
43126 000040A3 268A368400                   mov dh,[es:84h]
43127 000040A8 08F6                     or      dh,dh             ; # of rows valid?
43128 000040AA 7504                         jnz short menu_clear ; hopefully
43129 000040AC 8A36[804C]                   mov     dh,[bLastRow]   ; no, use a default
43130                                 menu_clear:
43131 000040B0 CD10                         int     10h             ; clear the screen using the req. attribute
43132 000040B2 07                           pop     es              ;
43133 000040B3 8836[804C]                   mov     [bLastRow],dh   ; save DH
43134 000040B7 BA[7752]                     mov     dx,_$MenuHeader
43135 000040BA E89709                       call    print           ; cursor now on row 3 (numbered from 0)
43136
43137 000040BD F606[814C]01                 test    byte [bDisableUI],1
43138 000040C2 751F                         jnz     short menu_nostatus
43139 000040C4 8A3E[7D4C]                   mov     bh,[bMenuPage]  ;
43140 000040C8 8A36[804C]                   mov     dh,[bLastRow]   ; restore DH
43141 000040CC B200                         mov     dl,0            ; print the status line on row DH, col 0,
43142 000040CE B402                         mov     ah,02h          ; now that we can trash the cursor position
43143 000040D0 CD10                         int     10h             ;
43144 000040D2 BA[C352]                     mov     dx,_$StatusLine
43145 000040D5 E87C09                       call    print           ;
43146 000040D8 B403                         mov     ah,3            ; get cursor position
43147 000040DA CD10                         int     10h             ;
43148 000040DC 80EA02                       sub     dl,2            ;
43149 000040DF 8816[7F4C]                   mov     [bLastCol],dl   ; save column where status char will go
43150
43151                                 menu_nostatus:
43152 000040E3 BB0100                       mov     bx,1            ; now prepare to display all the menu items
43153                                 menu_disploop:
43154 000040E6 E8B002                       call    print_item ; print item #BL
43155 000040E9 43                           inc     bx              ; why "inc bx"? because it's a 1-byte opcode
43156 000040EA 3A1E[874C]                   cmp     bl,[bMaxBlock]  ; all done?
43157 000040EE 76F6                         jbe short menu_disploop ; not yet
43158
43159                                 ;   Set cursor position to just below the menu items
43160
43161 000040F0 B200                         mov     dl,0            ; select column
43162 000040F2 88DE                         mov     dh,bl           ;
43163 000040F4 80C604                       add     dh,4            ; select row below menu
43164 000040F7 8A3E[7D4C]                   mov     bh,[bMenuPage]  ;
43165 000040FB B402                         mov     ah,02h          ; set cursor position beneath the block list
43166 000040FD CD10                         int     10h             ;
43167
43168 000040FF BA[B052]                     mov     dx,_$MenuPrmpt
43169 00004102 E84F09                       call    print           ;
43170 00004105 E82903                       call    select_item     ; make a selection, return # in BX
43171 00004108 BA[7050]                     mov     dx,crlfm
43172 0000410B E84609                       call    print           ;
43173 0000410E FF36[814C]                   push    word [bDisableUI]
43174 00004112 800E[814C]01                 or      byte [bDisableUI],1
43175 00004117 E86704                       call    show_status     ; clear the status line now
43176 0000411A 8F06[814C]                   pop     word [bDisableUI]
43177
43178                                 ;   Now begins the "re-organization" process...
43179
43180                                 menu_autoselect:
43181 0000411E 83FBFF                       cmp     bx,-1 ; 0FFFFh    ; clean boot requested?
43182 00004121 7508                         jne short normal_boot ; no
43183 00004123 E8F105                       call    disable_autoexec; basically, add a /D to the command.com line
43184                                 menu_abort:
43185 00004126 29C9                         sub     cx,cx           ; then immediately exit with 0 config.sys image
43186 00004128 E9E400                       jmp menu_exit     ;
43187
43188                                 normal_boot:
43189 0000412B 83FBFE                       cmp     bx,-2 ; 0FFFEh    ; back to top-level menu?
43190 0000412E 7509                         jne short not_topmenu ; no
43191 00004130 8B0E[5603]                   mov     cx,[count]      ; yes, start all over
43192 00004134 29F6                         sub     si,si           ;
43193 00004136 E9FFFD                       jmp     menu_search
43194
43195                                 not_topmenu:
43196 00004139 80BF[8C4C]4F                 cmp     byte [abBlockType+bx],CONFIG_SUBMENU
43197 0000413E 7510                         jne short not_submenu
43198 00004140 01DB                         add     bx,bx           ;
43199 00004142 8BBF[964C]                   mov     di,[aoffBlockName+bx]
43200 00004146 E8E101                       call    srch_block      ; THIS CANNOT FAIL!
43201 00004149 89FE                         mov     si,di           ;
43202 0000414B 89D9                         mov     cx,bx           ; ES:SI and CX are ready for another round
43203 0000414D E929FE                       jmp     menu_found
43204
43205                                 not_submenu:
43206 00004150 01DB                         add     bx,bx           ; get BX -> name of selected block
43207 00004152 8B9F[964C]                   mov     bx,[aoffBlockName+bx]
43208
43209                                 ;   BX should now either be ZERO (meaning no block has been selected) or
43210                                 ;   the offset relative to ES of the block name to be processed (along with
43211                                 ;   all the "common" lines of course)
43212
43213                                 no_selection:
43214 00004156 891E[884C]                   mov     [offDefBlock],bx; save selection
43215 0000415A 8B0E[5603]                   mov     cx,[count]      ; reset ES:SI and CX for reprocessing
43216 0000415E 29F6                         sub     si,si           ;
43217 00004160 1E                           push    ds              ;
43218 00004161 8E1E[6219]                   mov     ds,[config_wrkseg]; this is where we'll store new config.sys image
43219 00004165 29FF                         sub     di,di           ;
43220
43221                                 ;   ES:SI-> config.sys, DS:DI -> new config.sys workspace
43222                                 ;
43223                                 ;   Work our way through the config.sys image again, this time copying
43224                                 ;   all lines that are (A) "common" lines outside any block or (B) lines
43225                                 ;   within the requested block. Lines inside INCLUDEd blocks are transparently
43226                                 ;   copied by copy_block in a recursive fashion; the amount of recursion is
43227                                 ;   limited by the fact INCLUDE statements are REMed by copy_block as they are
43228                                 ;   processed and by the number of unique INCLUDE stmts in config.sys...
43229                                 ;
43230                                 ;   BUGBUG 20-Mar-1992 JeffPar: If we can figure out the lower bound of the
43231                                 ;   stack we're running on, then we should check it inside copy_block
43232
43233                                 copyblock_loop:
43234 00004167 53                           push    bx              ; save selected block name
43235 00004168 E82F01                       call    copy_block      ; process (named or common) block
43236 0000416B 5B                           pop     bx              ;
43237 0000416C 7232                         jc  short move_config ; hit eof
43238
43239                                 ;   copy_block can only return for two reasons: it hit eof or a new block
43240
```

```
43241                                   copyblock_begin:
43242
43243                                   ; 10/09/2023
43244                                   %if 0
43245                                           push    ax              ;
43246                                           push    cx              ;
43247                                           push    si              ;
43248                                           push    di              ; always do "common" blocks
43249                                           mov     di,szCommon
43250                                           push    ds              ;
43251                                           push    cs              ;
43252                                           pop     ds              ;
43253                                           call    comp_names      ;
43254                                           pop     ds              ;
43255                                           pop     di              ;
43256                                           pop     si              ;
43257                                           pop     cx              ;
43258                                           pop     ax              ;
43259                                           je  short copyblock_check
43260                                   %endif
43261                                       ; 10/09/2023
43262 0000416E 57                          push    di
43263 0000416F BF[CB4C]                    mov     di,szCommon    ; always do "common" blocks
43264 00004172 E81602                      call    comp_names_x   ; (comp_names_safe)
43265 00004175 5F                          pop     di
43266 00004176 740F                        je      short copyblock_check
43267
43268 00004178 09DB                            or      bx,bx           ; is there a block name to check?
43269 0000417A 7414                            jz  short copyblock_skip ; no
43270 0000417C 57                              push    di              ;
43271 0000417D 89DF                            mov     di,bx           ; check block against given block name
43272 0000417F 1E                              push    ds              ;
43273 00004180 06                              push    es              ;
43274 00004181 1F                              pop     ds              ;
43275 00004182 E8E501                          call    comp_names      ; is this the block we really want to do?
43276 00004185 1F                              pop     ds              ;
43277 00004186 5F                              pop     di              ;
43278                                   copyblock_check:
43279 00004187 7217                            jc  short move_config ; hit eof
43280 00004189 7505                            jne short copyblock_skip ;
43281 0000418B E85A04                          call    skip_opt_line   ;
43282 0000418E EBD7                            jmp short copyblock_loop
43283
43284                                   copyblock_skip:                 ;
43285 00004190 E85504                          call    skip_opt_line   ; this ain't the block we wanted, so skip it
43286 00004193 E87804                          call    get_char        ;
43287 00004196 7208                            jc  short move_config ; hit eof
43288 00004198 247F                            and     al,~CONFIG_OPTION_QUERY ; 7Fh
43289 0000419A 3C5B                            cmp     al,CONFIG_BEGIN ;
43290 0000419C 74D0                            je  short copyblock_begin
43291 0000419E EBF0                            jmp     short copyblock_skip ; anything else is just skipped
43292                                   ;
43293                                   ;   To create as little risk to the rest of SysInit as little as possible,
43294                                   ;   and to free the workspace at "config_wrkseg" for creating an environment,
43295                                   ;   copy the new config.sys image to "confbot"
43296                                   ;
43297                                   move_config:
43298 000041A0 89F9                            mov     cx,di           ; now copy workspace at DS:DI to "confbot"
43299 000041A2 51                              push    cx              ;
43300                                   ;
43301                                   ;   But first, copy the CONFIG=<configuration><0> string to the workspace,
43302                                   ;   since the configuration name only currently exists in the "confbot" area
43303                                   ;
43304                                       ;mov    cx,7
43305 000041A3 B90700                      mov     cx,szMenu-szBoot-1
43306 000041A6 BE[BE4C]                        mov     si,szBoot ; first copy the CONFIG= part
43307 000041A9 47                              inc     di              ; skip a byte, in case absolutely nothing
43308                                                                  ; was copied to the workspace, because we always
43309                                                                  ; zero the first byte of the workspace (below)
43310                                   copy_boot:
43311                                       ;lods    byte ptr cs:[si];
43312 000041AA 2E                              cs
43313 000041AB AC                          lodsb
43314 000041AC 8805                        mov     [di],al         ;
43315 000041AE 47                              inc     di              ;
43316 000041AF E2F9                            loop    copy_boot       ;
43317
43318 000041B1 06                              push    es              ; then copy the configuration name
43319                                           ;mov       cx,128-7      ; put an upper limit on the name, to be safe
43320                                       ; 04/01/2023
43321 000041B2 B179                        mov     cl,128-7
43322 000041B4 2E8B36[884C]                mov     si,[cs:offDefBlock]; ES:SI -> default block name
43323 000041B9 09F6                            or      si,si           ; valid?
43324 000041BB 7505                            jnz short l1       ; yes
43325 000041BD 0E                              push    cs              ;
43326 000041BE 07                              pop     es              ;
43327 000041BF BE[CB4C]                        mov     si,szCommon
43328 000041C2 268A04                  l1:     mov     al,[es:si]      ;
43329 000041C5 E8B205                          call    any_delim       ;
43330 000041C8 7406                            je  short l2       ;
43331 000041CA 8805                            mov     [di],al         ;
43332 000041CC 46                              inc     si              ;
43333 000041CD 47                              inc     di              ;
43334 000041CE E2F2                            loop    l1              ;
43335 000041D0 C6050A                  l2:     mov     byte [di],lf     ; terminate the configuration string
43336 000041D3 07                              pop     es              ;
43337
43338                                   ;   Now we can copy "config_wrkseg" (DS) to "confbot" (ES)
43339
43340 000041D4 29FF                            sub     di,di           ;
43341 000041D6 2E893E[6019]                    mov     [cs:config_envlen],di
43342 000041DB 29F6                            sub     si,si           ;
43343 000041DD 59                              pop     cx              ; recover the size of "config_wrkseg"
43344
43345 000041DE 51                              push    cx              ;
43346 000041DF F3A4                            rep     movsb           ; moved!
43347 000041E1 59                              pop     cx              ;
43348 000041E2 8CD8                            mov     ax,ds           ;
43349 000041E4 1F                              pop     ds              ;
43350
43351                                   ;   Now that the config_wrkseg is available once again, we shall
43352                                   ;   use it to create an environment. The first thing to go in will be
43353                                   ;   the "CONFIG=configuration" thing. It is also important to zero
43354                                   ;   the first byte of the workspace, so that copy_envvar knows the buffer
43355                                   ;   is empty.
43356
43357 000041E5 06                              push    es              ;
43358 000041E6 8EC0                            mov     es,ax           ;
43359 000041E8 46                              inc     si              ; ES:SI -> "CONFIG=configuration"
43360 000041E9 26C606000000                    mov     byte [es:0],0    ;empty the environment block
43361 000041EF E82600                          call    copy_envvar     ; copy envvar at ES:SI to "config_wrkseg"
43362 000041F2 07                              pop     es
43363
43364                                   ;   Before returning, restore the default video page setting but do NOT
```

```
43365                                   ;   do it using INT 10h's Set Active Page function, because if the menu was
43366                                   ;   displayed on a different page, then it's because we don't want to see
43367                                   ;   all the device driver/TSR goop (which goes to the default page)
43368
43369                                   menu_done:
43370 000041F3 803E[7D4C]00                       cmp     byte [bMenuPage],0
43371 000041F8 7415                               je  short menu_exit       ;
43372 000041FA 06                                 push    es              ;
43373 000041FB B84000                             mov     ax,40h          ;
43374 000041FE 8EC0                               mov     es,ax           ;
43375 00004200 A1[834C]                           mov     ax,[wCRTStart]  ;
43376 00004203 26A34E00                           mov     [es:004Eh],ax   ;
43377 00004207 A0[824C]                           mov     al,[bCRTPage]   ;
43378 0000420A 26A26200                           mov     [es:0062h],al   ;
43379 0000420E 07                                 pop     es              ;
43380                                   menu_exit:
43381 0000420F 890E[5603]                         mov     [count],cx      ; set new counts
43382 00004213 890E[5803]                         mov     [org_count],cx  ;
43383                                             ; 10/09/2023 (*) - Erdogan Tan
43384                                       ; MSDOS 6.21 IO.SYS - SYSINIT:46D3h
43385                                       ; PCDOS 7.1 IBMBIO.COM - SYSINIT:491Ah
43386                                       ;sub   si,si           ; always return ES:SI pointing to config.sys
43387 00004217 C3                         retn
43388
43389                                   ; (*) NOTE: MSDOS 6.0 source code (SYSINIT2.ASM) contains 'sub si,si' at this
43390                                   ;   position (then 'retn' just after it)
43391                                   ;   but MSDOS 6.21 and PCDOS 7.1 SYSINITs contain only 'retn' here.
43392
43393                                   ;--------------------------------------------------------------------------
43394                                   ;
43395                                   ;   copy_envvar:  copy the envvar at ES:SI to "config_wrkseg"
43396                                   ;
43397                                   ;   INPUT
43398                                   ;    ES:SI -> environment variable (in the form "var=string<cr/lf>")
43399                                   ;
43400                                   ;   OUTPUT
43401                                   ;       config_envlen (ie, where to put next envvar) updated appropriately
43402                                   ;       carry set if error (eg, missing =); clear otherwise
43403                                   ;
43404                                   ;   OTHER REGS USED
43405                                   ;       None
43406                                   ;
43407                                   ;   NOTES
43408                                   ;       None
43409                                   ;
43410                                   ;   HISTORY
43411                                   ;       Created 29-Mar-1992 by JeffPar
43412                                   ;
43413                                   ;--------------------------------------------------------------------------
43414
43415                                       ; 04/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
43416                                       ; (SYSINIT:46D4h)
43417
43418                                   copy_envvar:
43419 00004218 51                                 push    cx              ;
43420 00004219 56                                 push    si              ;
43421 0000421A 1E                                 push    ds              ;
43422 0000421B 06                                 push    es              ;
43423 0000421C 06                                 push    es              ;
43424 0000421D 8E06[6219]                         mov     es,[config_wrkseg] ; ES:DI to point to next available byte
43425 00004221 1F                                 pop     ds                 ; DS:SI to point to envvar
43426
43427                                   ;   Have to calculate the length of the variable name (and if we hit
43428                                   ;   the end of the line before we hit '=', then it's curtains for this
43429                                   ;   config.sys line)
43430                                   ;
43431                                   ;   The check for NULL is important because copy_envvar is also used to copy
43432                                   ;   the initial CONFIG= setting, which will have been zapped by a NULL if no
43433                                   ;   menu block existed (in order to prevent the creation of an environment)
43434
43435 00004222 29C9                             sub     cx,cx           ;
43436                                   copy_varlen:                      ;
43437 00004224 AC                                 lodsb                   ;
43438 00004225 08C0                               or      al,al           ; NULL?
43439                                             ;stc       ; 10/09/2023 (x)
43440 00004227 746B                               jz  short copy_envexit ; yes, abort
43441 00004229 3C0D                               cmp     al,cr           ;
43442                                             ;stc       ; 10/09/2023 (x)
43443 0000422B 7467                               je  short copy_envexit
43444 0000422D 3C0A                               cmp     al,lf           ;
43445                                             ;stc       ; 10/09/2023 (x)
43446 0000422F 7463                               je  short copy_envexit
43447 00004231 41                                 inc     cx              ;
43448 00004232 3C3D                               cmp     al,'='          ;
43449 00004234 75EE                               jne short copy_varlen
43450 00004236 B000                               mov     al,0            ;
43451 00004238 8A24                               mov     ah,[si]         ; save char after '='
43452 0000423A 29CE                               sub     si,cx           ; back up to given varname
43453 0000423C 49                                 dec     cx              ; CX == # of bytes in varname
43454 0000423D 29FF                               sub     di,di           ; start looking for DS:SI at ES:0
43455                                   copy_varsrch:
43456 0000423F 263805                             cmp     byte [es:di],al
43457 00004242 7425                               je  short copy_envprep ; search failed, just copy var
43458 00004244 89FB                               mov     bx,di           ; ES:BX -> start of this varname
43459 00004246 51                                 push    cx              ;
43460 00004247 56                                 push    si              ;
43461 00004248 F3A6                               repe    cmpsb           ;
43462 0000424A 5E                                 pop     si              ;
43463 0000424B 59                                 pop     cx              ;
43464 0000424C 7531                               jne short copy_varnext ; no match, skip to next varname
43465 0000424E 26803D3D                           cmp     byte [es:di],'='
43466 00004252 752B                               jne     short copy_varnext ; no match, there's more characters
43467
43468                                   ;   Previous occurrence of variable has been found; determine the
43469                                   ;   entire length and then destroy it
43470
43471 00004254 B9FFFF                             mov     cx,-1           ;
43472 00004257 F2AE                               repne   scasb           ; guaranteed to get null (since we put it there)
43473 00004259 56                                 push    si              ;
43474 0000425A 89FE                               mov     si,di           ;
43475 0000425C 89DF                               mov     di,bx           ;
43476 0000425E 2E8B0E[6019]                       mov     cx,[cs:config_envlen]
43477 00004263 29F1                               sub     cx,si           ; destroy variable now
43478                                         ;rep movs byte ptr es:[di],byte ptr es:[si]
43479                                         ;;db 0F3h,26h,0A4h ; MSDOS 6.21 IO.SYS - SYSINIT:4724h
43480
43481 00004265 F3                         rep     ; 0F3h
43482 00004266 26                         es      ; 26h
43483 00004267 A4                         movsb   ; 0A4h
43484
43485 00004268 5E                         pop     si
43486                                   copy_envprep:
43487 00004269 80FC0D                             cmp     ah,cr           ; if there is nothing after the '='
43488 0000426C 741D                               je  short copy_envdel ; then just exit with variable deleted
```

```
43489 0000426E 80FC0A                          cmp     ah,lf           ;
43490 00004271 7418                            je  short copy_envdel
43491                                          ;jmp       short copy_envloop
43492                                      ; 04/01/2023
43493                                  copy_envloop:                   ;
43494 00004273 AC                          lodsb                   ;
43495 00004274 3C0D                        cmp     al,cr           ;
43496 00004276 7410                        je      short copy_envdone
43497 00004278 3C0A                        cmp     al,lf           ;
43498 0000427A 740C                        je      short copy_envdone
43499 0000427C AA                          stosb                   ;
43500 0000427D EBF4                        jmp     short copy_envloop
43501
43502                                  copy_varnext:                   ;
43503 0000427F 51                          push    cx              ;
43504 00004280 B9FFFF                      mov     cx,-1           ;
43505 00004283 F2AE                        repne   scasb           ;
43506 00004285 59                          pop     cx              ;
43507 00004286 EBB7                        jmp short copy_varsrch
43508
43509                                      ; 04/01/2023
43510                                  ;copy_envloop:                  ;
43511                                  ;   lodsb                   ;
43512                                  ;   cmp     al,cr           ;
43513                                  ;   je      short copy_envdone
43514                                  ;   cmp     al,lf           ;
43515                                  ;   je      short copy_envdone
43516                                  ;   stosb                   ;
43517                                  ;   jmp     short copy_envloop
43518
43519                                  copy_envdone:                   ;
43520 00004288 28C0                        sub     al,al           ; do SUB to clear carry as well
43521 0000428A AA                          stosb                   ; always null-terminate these puppies
43522                                  copy_envdel:                    ;
43523 0000428B 2688 05                     mov     [es:di],al      ; and stick another null to terminate the env.
43524 0000428E 2E893E[6019]                mov     [cs:config_envlen],di
43525                                      ; 10/09/2023 (x) - Erdogan Tan
43526 00004293 F9                          stc ; in order to clear carry flag via cmc (compact code trick!)
43527                                  copy_envexit:                   ;
43528 00004294 F5                          cmc ; (x) ; reverse carry flag status (je -> cf=1)
43529 00004295 07                          pop     es              ;
43530 00004296 1F                          pop     ds              ;
43531 00004297 5E                          pop     si              ;
43532 00004298 59                          pop     cx              ;
43533
43534                                  copy_done:  ; 18/12/2022
43535 00004299 C3                          retn
43536
43537                                  ;--------------------------------------------------------------------------
43538                                  ;
43539                                  ;   copy_block:  copy the current block to the new config.sys workspace
43540                                  ;
43541                                  ;   INPUT
43542                                  ;       CX == remaining bytes in "organized" config.sys memory image
43543                                  ;    ES:SI -> remaining bytes in "organized" config.sys memory image
43544                                  ;    DS:DI -> new config.sys workspace (equal in size to the original
43545                                  ;             config.sys image) where the current block is to be copied
43546                                  ;
43547                                  ;   OUTPUT
43548                                  ;       Same as above
43549                                  ;       AL also equals the last character read from the organized image
43550                                  ;
43551                                  ;   OTHER REGS USED
43552                                  ;       All
43553                                  ;
43554                                  ;   NOTES
43555                                  ;       None
43556                                  ;
43557                                  ;   HISTORY
43558                                  ;       Created 16-Mar-1992 by JeffPar
43559                                  ;
43560                                  ;--------------------------------------------------------------------------
43561
43562                                      ; 04/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
43563                                      ; (SYSINIT:4759h)
43564
43565                                  copy_block:
43566 0000429A E87103                      call    get_char        ; check for include
43567 0000429D 72FA                        jc  short copy_done         ;
43568 0000429F 247F                    and     al,~CONFIG_OPTION_QUERY ; 7Fh
43569 000042A1 3C5B                    cmp     al,CONFIG_BEGIN ; another BEGIN implies END as well
43570 000042A3 74F4                        je  short copy_done ;
43571
43572 000042A5 3C4A                        cmp     al,CONFIG_INCLUDE ; 'J'
43573 000042A7 88E0                        mov     al,ah           ; AL == the original line code
43574 000042A9 753A                        jne short copy_line         ; not an "include" line
43575
43576                                  ;   We have hit an "INCLUDE" line; first, REM out the line so that we
43577                                  ;   never try to include the block again (no infinite include loops please),
43578                                  ;   then search for the named block and call copy_block again.
43579
43580 000042AB 26C644FF30                  mov     byte [es:si-1],CONFIG_REM ; '0'
43581 000042B0 57                          push    di              ;
43582
43583 000042B1 BF[C64C]                    mov     di,szMenu
43584 000042B4 E8D400                      call    comp_names_safe ; don't allow INCLUDE MENU
43585 000042B7 7426                        je  short copy_skip         ;
43586
43587 000042B9 BF[CB4C]                    mov     di,szCommon
43588 000042BC E8CC00                      call    comp_names_safe ; don't allow INCLUDE COMMON
43589 000042BF 741E                        je  short copy_skip         ;
43590
43591 000042C1 89F7                        mov     di,si           ; try to find the block
43592 000042C3 E86400                      call    srch_block      ;
43593 000042C6 89FA                        mov     dx,di           ;
43594                                  ; 10/09/2023
43595                                  ;pop    di              ;
43596 000042C8 7514                        jne short copy_error ; no such block
43597 000042CA 5F                      pop     di  ; 10/09/2023
43598 000042CB 51                          push    cx              ;
43599 000042CC 89D9                        mov     cx,bx           ;
43600 000042CE 56                          push    si              ;
43601 000042CF 4A                          dec     dx              ;
43602 000042D0 89D6                        mov     si,dx           ;
43603 000042D2 E80E03                      call    skip_line       ; skip the rest of the "block name" line
43604 000042D5 E8C2FF                      call    copy_block      ; and copy in the rest of that block
43605 000042D8 5E                          pop     si              ;
43606 000042D9 59                          pop     cx              ;
43607 000042DA 28C0                        sub     al,al           ; force skip_opt_line to skip...
43608 000042DC EB2B                        jmp     short copy_nextline
43609
43610                                  copy_error:
43611                                      ; 10/09/2023
43612 000042DE F8                          clc
```

```
43613                                   copy_skip:
43614 000042DF 5F                               pop     di
43615                                   ;copy_error:
43616                                       ; 10/09/2023 (cf=0)
43617                                       ;clc                    ;
43618 000042E0 E80004                           call    print_error     ; note that carry is clear, no pause
43619 000042E3 EB24                             jmp     short copy_nextline
43620
43621                                   ;   Copy the line at ES:SI to the current location at DS:DI
43622
43623                                   copy_line:
43624 000042E5 8805                             mov     [di],al         ;
43625 000042E7 47                               inc     di              ;
43626 000042E8 3C20                             cmp     al,' '          ; is this is a "real" line with a "real" code?
43627 000042EA 721D                             jb  short copy_nextline ; no
43628 000042EC 2E803E[6519]00                   cmp     byte [cs:config_multi],0
43629 000042F2 7409                             je  short copy_loop       ; not a multi-config config.sys, don't embed #s
43630 000042F4 E81700                           call    get_linenum       ; BX == line # of line @ES:SI
43631 000042F7 891D                             mov     [di],bx         ; stash it immediately following the line code
43632 000042F9 47                               inc     di              ;
43633 000042FA 47                               inc     di              ;
43634 000042FB EB08                             jmp     short copy_next ;
43635                                   copy_loop:                      ;
43636 000042FD E80E03                           call    get_char        ;
43637 00004300 7297                             jc  short copy_done ; end of file
43638 00004302 8805                             mov     [di],al         ;
43639 00004304 47                               inc     di              ;
43640                                   copy_next:
43641 00004305 3C0A                             cmp     al,lf ; 0Ah       ; done with line?
43642 00004307 75F4                             jne short copy_loop       ; nope
43643
43644                                   copy_nextline:
43645 00004309 E8DC02                           call    skip_opt_line   ;
43646 0000430C EB8C                             jmp     short copy_block

                                            ; 18/12/2022
                                      ;copy_done:
                                              ;retn
                                      ;---------------------------------------------------------------------------
                                      ;
                                      ;   get_linenum:  return line # (in BX) of current line (@ES:SI)
                                      ;
                                      ;   INPUT
                                      ;    ES:SI -> some line in the config.sys memory image
                                      ;
                                      ;   OUTPUT
                                      ;       BX == line # (relative to 1)
                                      ;
                                      ;   OTHER REGS USED
                                      ;       DX
                                      ;
                                      ;   NOTES
                                      ;       None
                                      ;
                                      ;   HISTORY
                                      ;       Created 16-Mar-1992 by JeffPar
                                      ;
                                      ;---------------------------------------------------------------------------

                                      get_linenum:
0000430E 50                                     push    ax              ;
0000430F 29DB                                   sub     bx,bx           ; BX == line # (to be returned)
00004311 51                                     push    cx              ;
00004312 89F2                                   mov     dx,si           ; DX == the offset we're looking for
00004314 56                                     push    si              ;
00004315 2E8B0E[5603]                           mov     cx,[cs:count]   ;
0000431A 29F6                                   sub     si,si           ; prepare to scan entire file
                                      get_linenum_loop:                 ;
0000431C E8C402                                 call    skip_line       ;
0000431F 7205                                   jc  short get_linenum_done
00004321 43                                     inc     bx              ;
00004322 39D6                                   cmp     si,dx           ; have we exceeded the desired offset yet?
00004324 72F6                                   jb  short get_linenum_loop ; no
                                      get_linenum_done:                 ;
00004326 5E                                     pop     si              ;
00004327 59                                     pop     cx              ;
00004328 58                                     pop     ax              ;
00004329 C3                                     retn

                                      ;---------------------------------------------------------------------------
                                      ;
                                      ;   srch_block:  searches entire config.sys for block name @ES:DI
                                      ;
                                      ;   INPUT
                                      ;       ES -> config.sys image
                                      ;    ES:DI -> block name to find
                                      ;
                                      ;   OUTPUT
                                      ;       ZF flag set, if found
                                      ;    ES:DI -> just past the name in the block heading, if found
                                      ;       BX == # bytes remaining from that point, if found
                                      ;
                                      ;   OTHER REGS USED
                                      ;       None
                                      ;
                                      ;   NOTES
                                      ;       This differs from "find_block" in that it searches the ENTIRE
                                      ;       config.sys image, not merely the remaining portion, and that it
                                      ;       takes a pointer to block name that is *elsewhere* in the image
                                      ;       (ie, ES) as opposed to some string constant in our own segment (DS).
                                      ;
                                      ;   HISTORY
                                      ;       Created 16-Mar-1992 by JeffPar
                                      ;
                                      ;---------------------------------------------------------------------------

                                      srch_block:           ; returns BX -> named block in CONFIG.SYS
0000432A 50                                     push    ax              ;
0000432B 51                                     push    cx              ;
0000432C 2E8B0E[5603]                           mov     cx,[cs:count]   ;
00004331 56                                     push    si              ;
00004332 29F6                                   sub     si,si           ;
00004334 1E                                     push    ds              ;
00004335 06                                     push    es              ;
00004336 1F                                     pop     ds              ;
00004337 E80900                                 call    find_block      ;
0000433A 89F7                                   mov     di,si           ;
0000433C 89CB                                   mov     bx,cx           ;
0000433E 1F                                     pop     ds              ;
0000433F 5E                                     pop     si              ;
00004340 59                                     pop     cx              ;
00004341 58                                     pop     ax              ;
                                      find_exit: ; 16/04/2019
```

```
00004342 C3                             retn                    ;

                                ;---------------------------------------------------------------------------
                                ;
                                ;   find_block:  searches rest of config.sys for block name @DS:DI
                                ;
                                ;   INPUT
                                ;    DS:DI -> block name to find
                                ;    ES:SI -> remainder of config.sys image
                                ;       CX == remaining size of config.sys image
                                ;
                                ;   OUTPUT
                                ;       ZF flag set, if found (also, CF set if EOF)
                                ;    ES:SI -> where the search stopped (at end of block name or EOF)
                                ;       CX == # bytes remaining from that point
                                ;
                                ;   OTHER REGS USED
                                ;       AX
                                ;
                                ;   NOTES
                                ;       This differs from "srch_block" in that it searches only the
                                ;       remaining portion of the config.sys image and leaves SI and CX
                                ;       pointing to where the search left off, and that it takes a pointer
                                ;       to search string in our own segment (DS:DI instead of ES:DI).
                                ;
                                ;   HISTORY
                                ;       Created 16-Mar-1992 by JeffPar
                                ;
                                ;---------------------------------------------------------------------------

                                find_block:
00004343 E8C802                         call    get_char        ; get line code
00004346 72FA                           jc  short find_exit       ; end of file
00004348 247F                           and     al,~CONFIG_OPTION_QUERY
0000434A 3C5B                           cmp     al,CONFIG_BEGIN ; beginning of a block?
0000434C 740C                           je  short check_line ; no
0000434E 3C4A                           cmp     al,CONFIG_INCLUDE
00004350 7513                           jne short next_line       ;
00004352 2E800E[6519]01                 or  byte [cs:config_multi],1
00004358 EB0B                           jmp     short next_line ;
                                check_line:
0000435A 2E800E[6519]01                 or      byte [cs:config_multi],1
00004360 E80700                         call    comp_names      ; compare block names
00004363 76DD                           jbe short find_exit       ; end of file, or names matched
                                next_line:
00004365 E88002                         call    skip_opt_line   ; no, so skip to next line
00004368 EBD9                           jmp short find_block  ;
                                ;find_exit:
                                ;   retn

                                ;---------------------------------------------------------------------------
                                ;
                                ;   comp_names:  compares keyword @DS:DI to position in config.sys @ES:SI
                                ;
                                ;   INPUT
                                ;    DS:DI -> keyword to compare
                                ;    ES:SI -> position in config.sys
                                ;       CX == remaining bytes in config.sys
                                ;
                                ;   OUTPUT
                                ;       ZF flag set, if match (also, CF set if EOF)
                                ;    ES:SI -> where the comparison stopped (at end of block name or EOF)
                                ;       CX == # bytes remaining from that point
                                ;
                                ;   OTHER REGS USED
                                ;       AX
                                ;
                                ;   NOTES
                                ;       None
                                ;
                                ;   HISTORY
                                ;       Created 16-Mar-1992 by JeffPar
                                ;
                                ;---------------------------------------------------------------------------

                                comp_names:
0000436A 57                             push    di              ;
                                comp_loop:                      ;
0000436B E8A002                         call    get_char        ;
0000436E 7210                           jc  short comp_exit       ;
00004370 E80704                         call    any_delim       ; is next character a delimiter?
00004373 8A25                           mov     ah,[di]         ; (get next character we're supposed to match)
00004375 740B                           je  short comp_almost ; yes, it *could* be a match
00004377 47                             inc     di                ;
00004378 25DFDF                         and     ax,~2020h ; 0DFDFh
                                                                ; BUGBUG -- assumes both names are alphanumeric -JTP
0000437B 38E0                           cmp     al,ah           ; match?
0000437D 74EC                           je  short comp_loop ; yes, keep looking at the characters
0000437F F8                             clc                     ; prevent erroneous eof indication: clear carry
                                comp_exit:                      ;
00004380 5F                             pop     di              ;
00004381 C3                             retn                      ;
                                comp_almost:                    ;
00004382 86E0                           xchg    al,ah           ; we don't know for sure if it's a match
00004384 E8F303                         call    any_delim       ; until we verify that the second string has
00004387 86E0                           xchg    al,ah           ; been exhausted also...
00004389 EBF5                           jmp     short comp_exit ; if we are, this call to any_delim will tell...

                                ;---------------------------------------------------------------------------

                                    ; 10/09/2023 - Retrodos v4.2 IO.SYS (Optimization)
                                comp_names_x:
                                    ;
                                comp_names_safe:
0000438B 50                             push    ax
0000438C 51                             push    cx
0000438D 56                             push       si
0000438E 1E                             push    ds
0000438F 0E                             push    cs
00004390 1F                             pop     ds
00004391 E8D6FF                         call    comp_names
00004394 1F                             pop     ds
00004395 5E                         pop     si
00004396 59                             pop     cx
00004397 58                             pop     ax
00004398 C3                             retn

                                ;---------------------------------------------------------------------------
                                ;
                                ;   print_item:  display menu item #BL
                                ;
                                ;   INPUT
                                ;       BL == menu item # to display
                                ;
```

```
43861                                   ;   OUTPUT
43862                                   ;       Menu item displayed, with appropriate highlighting if BL == bDefBlock
43863                                   ;
43864                                   ;   OTHER REGS USED
43865                                   ;       None
43866                                   ;
43867                                   ;   NOTES
43868                                   ;       This function saves/restores the current cursor position, so you
43869                                   ;       needn't worry about it.
43870                                   ;
43871                                   ;   HISTORY
43872                                   ;       Created 16-Mar-1992 by JeffPar
43873                                   ;
43874                                   ;--------------------------------------------------------------------------
43875
43876                                       ; 04/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
43877                                       ; (SYSINIT:485Ah)
43878
43879                                   print_item:       ; prints menu item #BL (1 to N)
43880 00004399 50                             push    ax              ;
43881 0000439A 53                             push    bx              ;
43882 0000439B 51                             push    cx              ;
43883 0000439C 52                             push    dx              ;
43884 0000439D 56                             push    si              ;
43885 0000439E B403                           mov     ah,03h          ; get cursor position
43886 000043A0 8A3E[7D4C]                     mov     bh,[bMenuPage]  ; always page zero
43887 000043A4 CD10                           int     10h             ; DH/DL = row/column
43888 000043A6 52                             push    dx              ; save it
43889 000043A7 B402                           mov     ah,02h          ; set cursor position
43890 000043A9 88DE                           mov     dh,bl           ;
43891 000043AB 80C603                         add     dh,3            ;
43892 000043AE B205                           mov     dl,5            ;
43893 000043B0 CD10                           int     10h             ; set cursor position for correct row/col
43894 000043B2 88D8                           mov     al,bl           ;
43895 000043B4 0430                           add     al,'0'          ; convert menu item # to ASCII digit
43896 000043B6 8A26[7C4C]                     mov     ah,[bMenuColor] ; normal attribute
43897 000043BA 3A1E[864C]                     cmp     bl,[bDefBlock]  ; are we printing the current block?
43898 000043BE 7510                           jne short print_other ; no
43899 000043C0 80CC70                         or      ah,70h          ; yes, set bgnd color to white
43900 000043C3 88E5                           mov     ch,ah           ;
43901 000043C5 B104                           mov     cl,4            ;
43902 000043C7 D2C5                           rol     ch,cl           ;
43903 000043C9 38E5                           cmp     ch,ah           ; are fgnd/bgnd the same?
43904 000043CB 7503                           jne short print_other ; no
43905 000043CD 80F408                         xor     ah,08h          ; yes, so modify the fgnd intensity
43906                                   print_other:                    ;
43907 000043D0 B700                           mov     bh,0            ;
43908 000043D2 01DB                           add     bx,bx           ;
43909 000043D4 8BBF[AA4C]                     mov     di,[aoffBlockDesc+bx]
43910 000043D8 88E3                           mov     bl,ah           ; put the attribute in the correct register now
43911 000043DA 8A3E[7D4C]                     mov     bh,[bMenuPage]  ; get correct video page #
43912 000043DE B409                           mov     ah,09h          ; write char/attr
43913 000043E0 B90100                         mov     cx,1            ;
43914 000043E3 CD10                           int     10h             ;
43915 000043E5 FEC2                           inc     dl              ; increment column
43916 000043E7 B402                           mov     ah,02h          ;
43917 000043E9 CD10                           int     10h             ;
43918                                         ;mov       ax,0900h+'.'    ;
43919 000043EB B82E09                         mov ax,092Eh
43920 000043EE CD10                       int     10h             ; display '.'
43921 000043F0 FEC2                           inc     dl              ; increment column
43922 000043F2 B402                           mov     ah,02h          ;
43923 000043F4 CD10                           int     10h             ;
43924                                         ;mov       ax,0900h+' '    ;
43925 000043F6 B82009                         mov ax,0920h
43926 000043F9 CD10                       int     10h             ; display ' '
43927 000043FB FEC2                           inc     dl              ; increment column
43928 000043FD B402                           mov     ah,02h          ;
43929 000043FF CD10                           int     10h             ;
43930 00004401 06                             push    es              ;
43931                                   print_loop:                     ;
43932 00004402 268A05                         mov     al,[es:di]; get a character of the description
43933 00004405 47                             inc     di              ;
43934 00004406 3C09                           cmp     al,TAB ; 9; substitute spaces for tabs
43935 00004408 7502                           jne short print_nontab ;
43936 0000440A B020                           mov     al,' '          ;
43937                                   print_nontab:                   ;
43938 0000440C 3C20                           cmp     al,' '          ;
43939 0000440E 7215                           jb  short print_done ; stop at the 1st character < space
43940 00004410 3C24                           cmp     al,'$'          ;
43941 00004412 7411                           je  short print_done ; also stop on $
43942 00004414 B409                           mov     ah,09h          ; display function #
43943 00004416 CD10                           int     10h             ;
43944 00004418 FEC2                           inc     dl              ; increment column
43945 0000441A 80FA4E                         cmp     dl,78           ; far enough?
43946 0000441D 7306                           jae short print_done ; yes
43947 0000441F B402                           mov     ah,02h          ;
43948 00004421 CD10                           int     10h             ;
43949 00004423 EBDD                           jmp short  print_loop
43950                                   print_done:                     ;
43951 00004425 07                             pop     es              ;
43952 00004426 5A                             pop     dx              ;
43953 00004427 B402                           mov     ah,02h          ;
43954 00004429 CD10                           int     10h             ; restore previous row/col
43955 0000442B 5E                             pop     si              ;
43956 0000442C 5A                             pop     dx              ;
43957 0000442D 59                             pop     cx              ;
43958 0000442E 5B                             pop     bx              ;
43959 0000442F 58                             pop     ax              ;
43960 00004430 C3                             retn                      ;
43961
43962                                   ;--------------------------------------------------------------------------
43963                                   ;
43964                                   ;   select_item:  wait for user to select menu item, with time-out
43965                                   ;
43966                                   ;   INPUT
43967                                   ;       None
43968                                   ;
43969                                   ;   OUTPUT
43970                                   ;       BX == menu item # (1-N), or -1 for clean boot
43971                                   ;       Selected menu item highlighted
43972                                   ;       Cursor positioned beneath menu, ready for tty-style output now
43973                                   ;
43974                                   ;   OTHER REGS USED
43975                                   ;       None
43976                                   ;
43977                                   ;   NOTES
43978                                   ;       None
43979                                   ;
43980                                   ;   HISTORY
43981                                   ;       Created 16-Mar-1992 by JeffPar
43982                                   ;
43983                                   ;--------------------------------------------------------------------------
43984
```

```
43985                                   select_item:              ; returns digit value in BX (trashes AX/CX/DX)
43986 00004431 8A1E[864C]                       mov     bl,[bDefBlock]  ; BL will be the default block #
43987 00004435 88D8                             mov     al,bl           ;
43988 00004437 E83701                           call    disp_num        ;
43989 0000443A E84401                           call    show_status     ; display current interactive status
43990 0000443D 803E[8A4C]FF                     cmp     byte [secTimeOut],-1
43991 00004442 7452                             je  short input_key      ; no time-out, just go to input
43992 00004444 B42C                             mov     ah,GET_TIME ; 2Ch
43993 00004446 CD21                             int     21h             ;
43994 00004448 88F7                             mov     bh,dh           ; BH = initial # of seconds
43995                                   check_time:
43996 0000444A A0[8A4C]                         mov     al,[secTimeOut] ;
43997 0000444D 2A06[8B4C]                       sub     al,[secElapsed] ;
43998 00004451 730D                             jae short show_time       ;
43999 00004453 800E[854C]02                     or      byte [bQueryOpt],2  ; disable all further prompting
44000 00004458 C606[8B4C]00                     mov     byte [secElapsed],0
44001 0000445D E9F600                           jmp select_done    ; time's up!
44002                                   show_time:
44003 00004460 53                               push    bx              ;
44004 00004461 88C3                             mov     bl,al           ; save # in BL
44005 00004463 8A3E[7D4C]                       mov     bh,[bMenuPage]  ;
44006 00004467 B403                             mov     ah,03h          ; get cursor position
44007 00004469 CD10                             int     10h             ;
44008 0000446B 52                               push    dx              ;
44009 0000446C 80C208                       add     dl,8            ; move cursor to the right
44010 0000446F B402                             mov     ah,02h          ; set cursor position
44011 00004471 CD10                             int     10h             ;
44012 00004473 BA[2753]                         mov     dx,_$TimeOut
44013 00004476 E8DB05                           call    print           ; print the "Time remaining: " prompt
44014 00004479 88D8                             mov     al,bl           ; recover # from BL
44015 0000447B 98                               cbw                     ; this works because AL is always <= 90
44016 0000447C B10A                             mov     cl,10           ;
44017 0000447E F6F1                             div     cl              ; AL = tens digit, AH = ones digit
44018 00004480 88E1                             mov     cl,ah           ;
44019 00004482 0430                             add     al,'0'          ;
44020 00004484 B40E                             mov     ah,0Eh          ;
44021 00004486 CD10                             int     10h             ; write TTY tens digit
44022 00004488 88C8                             mov     al,cl           ;
44023 0000448A 0430                             add     al,'0'          ;
44024 0000448C B40E                             mov     ah,0Eh          ;
44025 0000448E CD10                             int     10h             ; write TTY ones digit
44026 00004490 5A                               pop     dx
44027 00004491 B402                             mov     ah,02h          ; set cursor position back to where it was
44028 00004493 CD10                             int     10h             ;
44029 00004495 5B                               pop     bx              ;
44030                                   input_key:
44031 00004496 B406                             mov     ah,RAW_CON_IO ; 6
44032 00004498 B2FF                             mov     dl,0FFh         ; input request
44033 0000449A CD21                             int     21h             ;
44034 0000449C 751F                             jnz short got_key  ;
44035 0000449E 803E[8A4C]FF                     cmp     byte [secTimeOut],-1; is there a time-out?
44036 000044A3 74F1                             je  short input_key       ; no, just go back to input
44037 000044A5 B42C                             mov     ah,GET_TIME     ;
44038 000044A7 CD21                             int     21h             ; DH = seconds
44039 000044A9 88F4                             mov     ah,dh           ;
44040 000044AB 28FE                             sub     dh,bh           ; should generally be zero or one
44041 000044AD 88E7                             mov     bh,ah           ;
44042 000044AF 7302                             jnc short got_time ;
44043 000044B1 B601                             mov     dh,1            ; it wrapped back to zero, so assume one
44044                                   got_time:
44045 000044B3 08F6                             or      dh,dh           ; any change?
44046 000044B5 74DF                             jz  short input_key       ; no
44047 000044B7 0036[8B4C]                       add     [secElapsed],dh ;
44048 000044BB EB8D                             jmp short check_time ;
44049                                   got_key:
44050 000044BD 50                               push    ax              ;
44051 000044BE B8FFFF                           mov     ax,-1           ; zap both secTimeOut and secElapsed
44052 000044C1 8706[8A4C]                       xchg    [secTimeOut],ax
44053 000044C5 3CFF                             cmp     al,-1           ; was time-out already disabled?
44054 000044C7 740E                             je  short timeout_disabled ; yes
44055 000044C9 53                               push    bx              ; let's disable # seconds display
44056 000044CA B8200A                           mov     ax,0A20h        ; write multiple spaces
44057 000044CD 8B1E[7C4C]                       mov     bx,[bMenuColor]
44058 000044D1 B95000                           mov     cx,80           ; 80 of them, to be safe
44059 000044D4 CD10                             int     10h             ; to completely obliterate # seconds display
44060 000044D6 5B                               pop     bx              ;
44061
44062                                   timeout_disabled:
44063 000044D7 58                               pop     ax              ;
44064 000044D8 08C0                             or      al,al           ; extended key pressed?
44065 000044DA 755A                             jnz short normal_key ; no
44066 000044DC CD21                             int     21h             ; get the next part of the key then
44067 000044DE 74B6                             jz  short input_key       ; hmmm, what happened to the second part?
44068
44069 000044E0 3C48                             cmp     al,48h          ; up arrow?
44070 000044E2 7510                             jne short not_up   ; no
44071 000044E4 80FB01                           cmp     bl,1            ; are we as up as up can get?
44072 000044E7 76AD                             jbe short input_key       ; yes, ignore it
44073 000044E9 FE0E[864C]                       dec     byte [bDefBlock] ;
44074 000044ED E8A9FE                           call    print_item      ; re-print the current item
44075 000044F0 FECB                             dec     bl              ; and then print the new current item
44076 000044F2 EB12                             jmp     short print1
44077                                   not_up:
44078 000044F4 3C50                             cmp     al,50h          ; down arrow?
44079 000044F6 7518                             jne short not_down ; no
44080 000044F8 3A1E[874C]                       cmp     bl,[bMaxBlock]  ; are we as down as down can get?
44081 000044FC 7310                             jae short to_input_key ; yes, ignore it
44082 000044FE FE06[864C]                       inc     byte [bDefBlock] ;
44083 00004502 E894FE                           call    print_item      ; re-print the current item
44084 00004505 43                               inc     bx              ; and then print the new current item
44085                                   print1:
44086 00004506 88D8                         mov     al,bl           ;
44087                                   print2:
44088 00004508 E88EFE                       call    print_item      ;
44089 0000450B E86300                           call    disp_num        ;
44090                                   to_input_key:
44091 0000450E EB86                             jmp short input_key ; 10/09/2023
44092                                   not_down:
44093 00004510 F606[814C]01                     test    byte [bDisableUI],1
44094 00004515 75F7                             jnz short to_input_key ; don't allow F8 or F5
44095 00004517 3C42                             cmp     al,42h          ; F8 function key?
44096 00004519 750B                             jne short not_f8   ; no
44097 0000451B 8036[854C]01                     xor     byte [bQueryOpt],1
44098 00004520 E85E00                           call    show_status     ;
44099 00004523 E970FF                           jmp     input_key ;
44100                                   not_f8:
44101 00004526 3C3F                             cmp     al,3Fh          ; F5 function key?
44102 00004528 75E4                             jne short to_input_key ; no
44103                                       ; 02/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
44104                                       ; MSDOS 6.21 IO.SYS - SYSINIT:49EBh
44105                                       ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:4C32h)
44106 0000452A 800E[854C]04                 or      byte [bQueryOpt],4 ; no more queries
44107 0000452F BBFFFF                           mov     bx,-1           ; special return code (-1) indicating clean boot
44108 00004532 B020                             mov     al,' '          ; don't want to display anything really;
```

```
00004534 EB26                           jmp     short disp_input ; just want to display the cr/lf sequence...

                                    normal_key:
00004536 3C0D                           cmp     al,0Dh          ; Enter?
00004538 741C                           je  short select_done ; yes
0000453A 3C08                           cmp     al,08h          ; backspace?
0000453C 7504                           jne short not_backspace ; no
0000453E BBFEFF                         mov     bx,-2 ; 0FFFEh    ; yes, special return code
00004541 C3                             retn                      ;
                                    not_backspace:
00004542 2C30                           sub     al,'0'          ; is greater than '0'?
00004544 76C8                           jbe short to_input_key ; no
00004546 3A06[874C]                     cmp     al,[bMaxBlock]  ; is less than or equal to the maximum digit?
0000454A 77C2                           ja  short to_input_key ; no
0000454C A2[864C]                       mov [bDefBlock],al  ;
0000454F E847FE                         call    print_item      ; redisplay the current selection
00004552 88C3                           mov     bl,al           ; set new selection
00004554 EBB2                           jmp short print2

                                    select_done:
00004556 B700                           mov     bh,0            ; return a full 16-bit value (for indexing)
00004558 88D8                           mov     al,bl           ;
0000455A 0430                           add     al,'0'          ; convert it into a digit, then display it

                                        ; fall into disp_input

                                    ; 16/04/2019 - Retro DOS v4.0

                                    ;--------------------------------------------------------------------------
                                    ;
                                    ;   disp_input:  display a single character + cr/lf
                                    ;
                                    ;   INPUT
                                    ;       AL == character to display
                                    ;
                                    ;   OUTPUT
                                    ;       None
                                    ;
                                    ;   OTHER REGS USED
                                    ;       None
                                    ;
                                    ;   NOTES
                                    ;       This function is used not only for the menu input selection but
                                    ;       also for the interactive line prompting (the y/n/a thing).
                                    ;
                                    ;   HISTORY
                                    ;       Created 16-Mar-1992 by JeffPar
                                    ;
                                    ;--------------------------------------------------------------------------


                                    disp_input:
0000455C 50                             push    ax
                                        ;cmp    al,' '
                                        ;jae    short disp_ok
                                        ;mov    al,' '
                                        ; 10/09/2023 - Retro DOS v4.2 IO:SYS (Optimization)
0000455D B220                           mov     dl,' ' ; 20h
0000455F 38D0                           cmp     al,dl
00004561 7602                           jna     short disp_input_ok
                                    disp_ok:
00004563 88C2                           mov     dl,al
                                    disp_input_ok:
00004565 B402                           mov     ah,STD_CON_OUTPUT ; 2
00004567 CD21                           int     21h
00004569 BA[7050]                       mov     dx,crlfm
0000456C E8E504                         call    print
0000456F 58                             pop     ax
00004570 C3                             retn

                                    ;--------------------------------------------------------------------------

                                    disp_num:
00004571 53                             push    bx
00004572 0430                           add     al,'0'
00004574 B40A                           mov     ah,0Ah
00004576 8B1E[7C4C]                     mov     bx,[bMenuColor]
0000457A B90100                         mov     cx,1
0000457D CD10                           int     10h
0000457F 5B                             pop     bx
00004580 C3                             retn

                                    ;--------------------------------------------------------------------------
                                    ;
                                    ;   show_status:  display current interactive mode setting (on/off/none)
                                    ;
                                    ;   INPUT
                                    ;       None
                                    ;
                                    ;   OUTPUT
                                    ;       None
                                    ;
                                    ;   OTHER REGS USED
                                    ;       None
                                    ;
                                    ;   NOTES
                                    ;       None
                                    ;
                                    ;   HISTORY
                                    ;       Created 16-Mar-1992 by JeffPar
                                    ;
                                    ;--------------------------------------------------------------------------

                                    show_status:
00004581 53                             push    bx              ; BL = video page #
00004582 8B1E[7C4C]                     mov     bx,[bMenuColor]
00004586 B403                           mov     ah,03h          ; get cursor position
00004588 CD10                           int     10h             ;
0000458A 52                             push    dx              ; save it
0000458B B402                           mov     ah,02h          ; set cursor position
0000458D 8B16[7F4C]                     mov     dx,[bLastCol]   ; set correct row/col
00004591 F606[814C]01                   test    byte [bDisableUI],1
00004596 740C                           jz  short show_onoff ; just show on/off
00004598 B200                           mov     dl,0            ;
0000459A CD10                           int     10h             ;
0000459C B8200A                         mov     ax,0A20h        ; write multiple spaces
0000459F B95000                         mov     cx,80           ; 80 of them, to be exact
                                        ; 10/09/2023
                                        ;int    10h             ; to obliterate the status line
000045A2 EB11                           jmp     short show_done ;
                                    show_onoff:
000045A4 CD10                           int     10h
                                            ; - VIDEO - WRITE CHARACTERS ONLY AT CURSOR POSITION
                                            ; AL = character, BH = display page - alpha mode
```

```
                                        ; BL = color of character (graphics mode, PCjr only)
                                        ; CX = number of times to write character
                                
000045A6 A0[2353]                       mov     al,[_$NO] ; assume OFF
000045A9 803E[854C]01                   cmp     byte [bQueryOpt],1 ; is interactive mode on?
000045AE 7503                           jne short show_noton ; no
000045B0 A0[1F53]                       mov     al,[_$YES] ; yes
                                show_noton:                     ;
000045B3 B40E                           mov     ah,0Eh          ; write TTY
                                show_done:  ; 10/09/2023
000045B5 CD10                           int     10h             ;
                                ;show_done:                     ;
000045B7 5A                             pop     dx              ;
000045B8 B402                           mov     ah,02h          ;
000045BA CD10                           int     10h             ; restore original cursor position
000045BC 5B                             pop     bx              ;
000045BD C3                             retn                     ;
                                
                                ; 16/04/2019 - Retro DOS v4.0
                                
                                ;--------------------------------------------------------------------------
                                ;
                                ;   skip_token: advances ES:SI/CX past the current token
                                ;
                                ;   INPUT
                                ;    ES:SI -> position in config.sys
                                ;       CX == remaining bytes in config.sys
                                ;
                                ;   OUTPUT
                                ;       CF set if EOL/EOF hit
                                ;       AL == 1st char of delimiter
                                ;    ES:SI -> just past the delimiter
                                ;       CX == # bytes remaining from that point
                                ;
                                ;   OTHER REGS USED
                                ;       AX
                                ;
                                ;   NOTES
                                ;       None
                                ;
                                ;   HISTORY
                                ;       Created 16-Mar-1992 by JeffPar
                                ;
                                ;--------------------------------------------------------------------------
                                
                                skip_token:
000045BE E84D00                         call    get_char
000045C1 7210                           jc  short skip_token_done
000045C3 E8B401                         call    any_delim
000045C6 75F6                           jne short skip_token
                                skip_check_eol:
000045C8 3C0D                           cmp     al,cr ; 0Dh
000045CA 7406                           je  short skip_token_eol
000045CC 3C0A                           cmp     al,lf ; 0Ah
000045CE 7402                           je  short skip_token_eol
000045D0 F8                             clc
                                        ;jmp       short skip_token_done
000045D1 C3                         retn
                                skip_token_eol:
000045D2 F9                             stc
                                skip_token_done:
000045D3 C3                             retn
                                
                                ;--------------------------------------------------------------------------
                                ;
                                ;   skip_delim: advances ES:SI/CX past the current delimiter
                                ;
                                ;   INPUT
                                ;    ES:SI -> position in config.sys
                                ;       CX == remaining bytes in config.sys
                                ;
                                ;   OUTPUT
                                ;       CF set if EOF hit
                                ;       AL == 1st char of token
                                ;    ES:SI -> just past the token
                                ;       CX == # bytes remaining from that point
                                ;    ES:BX -> new token (since ES:SI is already pointing 1 byte past token)
                                ;
                                ;   OTHER REGS USED
                                ;       AX
                                ;
                                ;   NOTES
                                ;       None
                                ;
                                ;   HISTORY
                                ;       Created 16-Mar-1992 by JeffPar
                                ;
                                ;--------------------------------------------------------------------------
                                
                                skip_delim: ; returns carry set if eol/eof
000045D4 E83700                         call    get_char        ;
000045D7 8D5CFF                         lea     bx,[si-1]       ; also returns BX -> next token
000045DA 72F7                           jc  short skip_token_done ;
000045DC E8AB01                         call    delim           ;
000045DF 74F3                           je  short skip_delim ;
000045E1 EBE5                           jmp short skip_check_eol  ; 13/05/2019
                                
                                ;--------------------------------------------------------------------------
                                ;
                                ;   skip_opt_line: same as skip_line provided AL != LF
                                ;
                                ;   INPUT
                                ;       AL == last character read
                                ;    ES:SI -> position in config.sys
                                ;       CX == remaining bytes in config.sys
                                ;
                                ;   OUTPUT
                                ;       CF set if EOF hit
                                ;       AL == 1st char of new line
                                ;    ES:SI -> just past 1st char of new line
                                ;       CX == # bytes remaining from that point
                                ;
                                ;   OTHER REGS USED
                                ;       AX
                                ;
                                ;   NOTES
                                ;       In other words, the purpose here is to skip to the next line,
                                ;       unless ES:SI is already sitting at the front of the next line (which
                                ;       it would be if the last character fetched -- AL -- was a linefeed)
                                ;
                                ;   HISTORY
                                ;       Created 16-Mar-1992 by JeffPar
                                ;
                                ;--------------------------------------------------------------------------
```

```
                                ; 03/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                ;skip_opt_line:
                                ;   cmp     al,lf ; 0Ah
                                ;   je      short skip_line_done
                                
                                    ; fall into skip_line
                                
                                ;--------------------------------------------------------------------------
                                ;
                                ;   skip_line: skip to the next line
                                ;
                                ;   INPUT
                                ;    ES:SI -> position in config.sys
                                ;       CX == remaining bytes in config.sys
                                ;
                                ;   OUTPUT
                                ;       CF set if EOF hit
                                ;    ES:SI -> just past 1st char of new line
                                ;       CX == # bytes remaining from that point
                                ;
                                ;   OTHER REGS USED
                                ;       AX
                                ;
                                ;   NOTES
                                ;       None
                                ;
                                ;   HISTORY
                                ;       Created 16-Mar-1992 by JeffPar
                                ;
                                ;--------------------------------------------------------------------------
                                
                                skip_line:
000045E3 E82800                         call    get_char
000045E6 7204                           jc  short skip_line_done
                                skip_opt_line:      ; 03/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
000045E8 3C0A                           cmp     al,lf ; 0Ah
000045EA 75F7                           jne short skip_line
                                skip_line_done:
                                num_done:   ; 18/12/2022
000045EC C3                             retn
                                
                                ;--------------------------------------------------------------------------
                                ;
                                ;   get_number: return binary equivalent of numeric string
                                ;
                                ;   INPUT
                                ;    ES:SI -> position in config.sys
                                ;       CX == remaining bytes in config.sys
                                ;
                                ;   OUTPUT
                                ;       AL == non-digit encountered
                                ;       BX == binary #
                                ;    ES:SI -> just past 1st non-digit
                                ;       CX == # bytes remaining from that point
                                ;
                                ;   OTHER REGS USED
                                ;       AX
                                ;
                                ;   NOTES
                                ;       None
                                ;
                                ;   HISTORY
                                ;       Created 16-Mar-1992 by JeffPar
                                ;
                                ;--------------------------------------------------------------------------
                                
                                ; 13/05/2019
                                
                                get_number:
000045ED 29DB                           sub     bx,bx           ; BX = result
                                num_loop:
000045EF E81C00                         call    get_char        ;
000045F2 72F8                           jc  short num_done ;
000045F4 3C30                           cmp     al,'0'          ; convert to value
000045F6 72F4                           jb  short num_done ; no more number
000045F8 3C39                           cmp     al,'9'          ;
000045FA 77F0                           ja  short num_done ;
000045FC 50                             push    ax              ;
000045FD B80A00                         mov     ax,10           ;
00004600 52                             push    dx              ;
00004601 F7E3                           mul     bx              ;
00004603 5A                             pop     dx              ;
00004604 89C3                           mov     bx,ax           ;
00004606 58                             pop     ax              ;
00004607 2C30                           sub     al,'0'          ;
00004609 98                             cbw                     ;
0000460A 01C3                           add     bx,ax           ;
0000460C EBE1                           jmp short num_loop ;
                                
                                    ; 18/12/2022
                                ;num_done:
                                        ;retn
                                
                                ;--------------------------------------------------------------------------
                                ;
                                ;   get_char:  return next character, advance ES:SI, and decrement CX
                                ;
                                ;   INPUT
                                ;    ES:SI -> position in config.sys
                                ;       CX == remaining bytes in config.sys
                                ;
                                ;   OUTPUT
                                ;       AL == next character
                                ;    ES:SI -> just past next character
                                ;       CX == # bytes remaining from that point
                                ;
                                ;   OTHER REGS USED
                                ;       AX
                                ;
                                ;   NOTES
                                ;       None
                                ;
                                ;   HISTORY
                                ;       Created 16-Mar-1992 by JeffPar
                                ;
                                ;--------------------------------------------------------------------------
                                
                                get_char:
0000460E 83E901                         sub     cx,1            ; use SUB to set carry,zero
00004611 7205                           jb  short get_fail ; out of data
                                        ;lods      byte ptr es:[si] ;
00004613 26                         es
00004614 AC                         lodsb
```

```
44481 00004615 88C4                     mov     ah,al           ;
44482 00004617 C3                       retn                      ;
44483                                 get_fail:                       ; restore CX to zero
44484 00004618 B90000                   mov     cx,0            ; leave carry set, zero not set
44485                                 nearby_ret:
44486 0000461B C3                       retn
44487
44488                                 ;--------------------------------------------------------------------------
44489                                 ;
44490                                 ;   query_user: ask user whether to execute current config.sys command
44491                                 ;
44492                                 ;   INPUT
44493                                 ;       AL == current command code
44494                                 ;    ES:SI -> current command line in config.sys
44495                                 ;    config_cmd == current command code, but with QUERY bit intact
44496                                 ;                  (00h used to generate "Process AUTOEXEC.BAT" prompt)
44497                                 ;
44498                                 ;   OUTPUT
44499                                 ;       CF set if command should be ignored (it is also REM'ed out)
44500                                 ;
44501                                 ;   OTHER REGS USED
44502                                 ;       BX, CX, DX, DI
44503                                 ;
44504                                 ;   NOTES
44505                                 ;       None
44506                                 ;
44507                                 ;   HISTORY
44508                                 ;       Created 16-Mar-1992 by JeffPar
44509                                 ;
44510                                 ;--------------------------------------------------------------------------
44511
44512                                     ; 31/12/2022 - Retro UNIX 386 v4.2 (Modified MSDOS 6.21 IO.SYS)
44513                                     ; (SYSINIT:4AE5h)
44514
44515                                     ; 12/12/2022
44516                                 query_user:
44517 0000461C F606[854C]04             test    byte [bQueryOpt],4      ; answer no to everything?
44518                                     ; 01/01/2023
44519 00004621 7403                     jz      short qu_1              ;
44520 00004623 E9B900                   jmp     skip_all
44521                                     ; 12/12/2022
44522                                     ;;jmp   short skip_all          ;
44523                                     ;jnz    short skip_all
44524                                 qu_1:
44525 00004626 F606[854C]02             test    byte [bQueryOpt],2      ; answer yes to everything?
44526 0000462B 75EE                     jnz short nearby_ret            ; yes (and return carry clear!)
44527 0000462D 50                       push    ax                      ;
44528 0000462E A0[6419]                 mov     al,[config_cmd]         ;
44529 00004631 F606[854C]01             test    byte [bQueryOpt],1      ; query every command?
44530 00004636 7506                     jnz short query_all             ; yes
44531 00004638 A880                     test    al,CONFIG_OPTION_QUERY  ;
44532                                     ; 01/01/2023
44533 0000463A 7502                     jnz     short query_all         ;
44534                                     ; 12/12/2022
44535                                     ;;jmp   short do_cmd            ;
44536                                     ;jz     short do_cmd ; cf=0
44537
44538                                     ; 01/01/2023
44539 0000463C 58                       pop     ax
44540 0000463D C3                       retn
44541
44542                                 query_all:
44543
44544                                 ;   Search for the command code (AL) in "comtab", and then print
44545                                 ;   out the corresponding keyword, followed by the rest of the actual
44546                                 ;   line pointed to by ES:SI
44547
44548 0000463E 56                       push    si                      ; save pointer to rest of CONFIG.SYS line
44549 0000463F BA[8253]                 mov     dx,_$AutoPrmpt          ;
44550 00004642 247F                     and     al,~CONFIG_OPTION_QUERY ; 7Fh
44551 00004644 7450                     jz  short generic_prompt  ; config_cmd must have been 0
44552
44553 00004646 88C6                     mov     dh,al                   ; save config_cmd in DH
44554 00004648 29DB                     sub     bx,bx                   ;
44555 0000464A BF[D24C]                 mov     di,comtab           ;
44556                                 find_match:                     ;
44557 0000464D 8A1D                     mov     bl,[di]                 ; get size of current keyword
44558 0000464F 08DB                     or      bl,bl                   ;
44559 00004651 7425                     jz  short line_print      ; end of table
44560 00004653 47                       inc     di                      ;
44561 00004654 3A01                     cmp     al,[di+bx]              ; match?
44562 00004656 7405                     je  short cmd_match             ; yes
44563 00004658 8D7901                   lea     di,[di+bx+1]            ; otherwise, skip this command code
44564                                     ; 13/05/2019
44565 0000465B EBF0                     jmp short find_match      ; loop
44566                                 cmd_match:                      ;
44567 0000465D 8A4DFF                   mov     cl,[di-1]               ;
44568 00004660 B500                     mov ch,0                        ;
44569 00004662 B402                     mov     ah,STD_CON_OUTPUT ; 2
44570                                 cmd_print:                      ;
44571 00004664 8A05                     mov     al,[di]                 ;
44572 00004666 47                       inc     di                      ;
44573 00004667 88C2                     mov     dl,al                   ;
44574 00004669 CD21                     int     21h                     ;
44575 0000466B E2F7                     loop    cmd_print               ;
44576 0000466D B23D                     mov     dl,'='                  ;
44577 0000466F 80FE56                   cmp     dh,CONFIG_SET  ; 'V'    ; for SET commands, don't display a '='
44578 00004672 7502                     jne short cmd_notset            ;
44579 00004674 B220                     mov     dl,' '                  ;
44580                                 cmd_notset:
44581 00004676 CD21                     int     21h                     ; '=' looks funny on SET commands
44582                                 line_print:                     ;
44583                                     ;lods   byte ptr es:[si]        ;
44584 00004678 26                       es
44585 00004679 AC                       lodsb
44586 0000467A 08C0                     or       al,al                  ;
44587 0000467C 7502                     jnz short non_null              ;
44588 0000467E B020                     mov     al,' '                  ;
44589                                 non_null:                       ;
44590 00004680 3C20                     cmp     al,' '                  ; control code?
44591 00004682 720F                     jb  short prompt_user     ; yes, assume end of line
44592 00004684 7505                     jne short non_space              ;
44593                                   ; 10/09/2023
44594 00004686 263804                   cmp     [es:si],al ; 20h
44595                                   ;cmp    byte [es:si],' '        ;
44596 00004689 7208                     jb  short prompt_user     ;
44597                                 non_space:                      ;
44598 0000468B 88C2                     mov     dl,al                   ;
44599 0000468D B402                     mov     ah,STD_CON_OUTPUT ; 2    ;
44600 0000468F CD21                     int     21h                     ;
44601 00004691 EBE5                     jmp short line_print      ;
44602
44603                                 prompt_user:                    ;
44604 00004693 BA[1353]                 mov     dx,_$InterPrmpt          ;
```

```
44605
44606                                   generic_prompt:
44607 00004696 E8BB03                           call    print                   ;
44608                                   input_loop:                             ;
44609 00004699 B400                             mov     ah,0                    ; read a key
44610 0000469B CD16                             int     16h                     ;
44611 0000469D 08C0                             or      al,al                   ; is it a function key?
44612 0000469F 750F                             jnz short not_func      ; no
44613 000046A1 80FC3F                           cmp     ah,3Fh                  ; F5 function key?
44614 000046A4 75F3                             jne short input_loop    ; no
44615 000046A6 A0[2353]                         mov     al,[_$NO]                ;
44616 000046A9 800E[854C]04                     or      byte [bQueryOpt],4       ; no more queries
44617 000046AE EB21                             jmp     short legal_char        ;
44618                                   not_func:
44619 000046B0 24DF                             and     al,~20h ; 0DFh           ; converting to upper case
44620 000046B2 3A06[2353]                       cmp     al,[_$NO]       ; verify character is legal
44621 000046B6 7419                             je  short legal_char    ;
44622 000046B8 3A06[1F53]                       cmp     al,[_$YES]      ;
44623 000046BC 7413                             je  short legal_char    ;
44624 000046BE 803E[6419]00                     cmp     byte [config_cmd],0      ;
44625 000046C3 74D4                             je  short input_loop    ; don't allow Esc on this query
44626 000046C5 3C1B                             cmp     al,1Bh                  ; Esc?
44627 000046C7 75D0                             jne short input_loop    ;
44628 000046C9 800E[854C]02                     or      byte [bQueryOpt],2       ; no more interactive boot prompts
44629 000046CE A0[1F53]                         mov     al,[_$YES]
44630                                   legal_char:                             ;
44631 000046D1 E888FE                           call    disp_input              ;
44632 000046D4 5E                               pop     si                      ; restore pointer to rest of CONFIG.SYS line
44633
44634 000046D5 3A06[2353]                       cmp     al,[_$NO]       ; process line?
44635 000046D9 7403                             je  short skip_cmd      ; no
44636                                       ; 12/12/2022
44637 000046DB F8                           clc
44638                                   do_cmd:
44639 000046DC 58                           pop     ax                      ;
44640                                       ; 12/12/2022
44641                                       ; cf=0
44642                                       ;clc                        ; just do the command
44643 000046DD C3                           retn
44644
44645                                   skip_cmd:
44646 000046DE 58                           pop     ax                      ;
44647                                   skip_all:
44648 000046DF B430                         mov     ah,CONFIG_REM ; '0'   ; fake out the rest of sysinit's processing
44649 000046E1 F9                           stc
44650 000046E2 C3                           retn
44651
44652                                   ;--------------------------------------------------------------------------
44653                                   ;
44654                                   ;   print_error: displays multi-config error conditions
44655                                   ;
44656                                   ;   INPUT
44657                                   ;    Carry set to pause, clear to not
44658                                   ;    ES:SI -> current command line in config.sys
44659                                   ;
44660                                   ;   OUTPUT
44661                                   ;       None
44662                                   ;
44663                                   ;   OTHER REGS USED
44664                                   ;       None
44665                                   ;
44666                                   ;   NOTES
44667                                   ;       None
44668                                   ;
44669                                   ;   HISTORY
44670                                   ;       Created 16-Mar-1992 by JeffPar
44671                                   ;
44672                                   ;--------------------------------------------------------------------------
44673
44674                                   print_error:
44675 000046E3 50                           push    ax
44676 000046E4 53                           push    bx
44677 000046E5 51                           push    cx
44678 000046E6 52                           push    dx
44679 000046E7 1E                           push    ds
44680 000046E8 0E                           push    cs
44681 000046E9 1F                           pop     ds
44682 000046EA 9C                           pushf
44683 000046EB E820FC                       call    get_linenum
44684 000046EE 891E[AF02]                   mov     [linecount],bx
44685 000046F2 E8C5E7                       call    error_line
44686 000046F5 9D                           popf
44687 000046F6 7319                         jnc short pe_ret
44688 000046F8 BA[DD51]                     mov     dx,_$PauseMsg
44689 000046FB E85603                       call    print
44690 000046FE B8070C                       mov     ax,0C07h            ; flush input buffer, then wait for key
44691 00004701 CD21                         int     21h                         ; wait for a key
44692 00004703 08C0                         or      al,al                       ; extended key?
44693 00004705 7504                         jnz short pe_1              ; no
44694 00004707 B407                         mov     ah,07h                      ; yes
44695 00004709 CD21                         int     21h                         ; eat it too
44696                                   pe_1:
44697 0000470B BA[7050]                     mov     dx,crlfm
44698 0000470E E84303                       call    print
44699                                   pe_ret:
44700 00004711 1F                           pop     ds
44701 00004712 5A                           pop     dx
44702 00004713 59                           pop     cx
44703 00004714 5B                           pop     bx
44704 00004715 58                           pop     ax
44705 00004716 C3                           retn
44706
44707                                   ;--------------------------------------------------------------------------
44708
44709                                   ;   This function is very simple: it merely prepends a "/D" to the
44710                                   ;   command-line for the shell; this (undocumented) switch disables
44711                                   ;   AUTOEXEC.BAT processing and the date/time prompt that is usually
44712                                   ;   displayed when there's no AUTOEXEC.BAT.
44713
44714                                   disable_autoexec:
44715                                       ; MSDOS 6.21 IO.SYS -  SYSINIT:4BE2h
44716                                       ; 17/04/2019 - Retro DOS v4.0
44717
44718 00004717 F606[854C]04                   test    byte [bQueryOpt],4
44719 0000471C 7443                         jz      short disable_exit
44720 0000471E F606[7B4C]01                 test    byte [dae_flag],1
44721 00004723 753C                         jnz     short disable_exit
44722 00004725 800E[7B4C]01                 or      byte [dae_flag],1
44723                                           ;or byte [bQueryOpt],2 ; MSDOS 6.0
44724 0000472A 810E[854C]0201               or       word [bQueryOpt],102h ; [bDefBlock] = 1
44725 00004730 BA4420                       mov      dx,'D ' ; 2044h
44726                                   dae_1:
44727                                       ; 03/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
44728 00004733 A0[BA4B]                         mov al,[def_swchr]
```

```
44729                                   ;mov    al,[command_line-1]     ; get default switchchar
44730 00004736 08C0                         or      al,al                   ; anything there?
44731 00004738 7427                         jz  short disable_exit    ; no, disable_autoexec already called
44732 0000473A 8A1E[BB4B]                   mov     bl,[command_line]       ;
44733 0000473E B700                         mov     bh,0                    ; BX == command-line length
44734 00004740 89D9                         mov     cx,bx                   ;
44735 00004742 80C303                       add     bl,3                    ;
44736 00004745 80FB7E                       cmp     bl,126                  ;
44737 00004748 7717                         ja  short disable_exit    ;
44738 0000474A 881E[BB4B]                   mov     [command_line],bl       ; update length
44739 0000474E 81C3[BC4B]                   add     bx,command_line+1 ; make sure we move the NULL too
44740 00004752 41                           inc     cx                      ; (just for consistency sake)
44741                                   disable_loop:                       ;
44742 00004753 8A67FD                       mov     ah,[bx-3]               ;
44743 00004756 8827                         mov     [bx],ah                 ;
44744 00004758 4B                           dec     bx                      ;
44745 00004759 E2F8                         loop    disable_loop            ;
44746 0000475B 8847FE                       mov     [bx-2],al               ;
44747                                   ;mov   word [bx-1],'D ' ; 2044h ; /D is stuffed into place now
44748 0000475E 8957FF                   mov    [bx-1],dx  ; MSDOS 6.21 IO.SYS - SYSINIT:4C29h
44749                                       ;mov       byte [command_line-1],0 ;
44750                                   disable_exit:                       ;
44751 00004761 C3                           retn
44752
44753                                   CheckQueryOpt:     ; MSDOS 6.21 IO.YSYS - SYSINIT:4C2Dh
44754 00004762 803E[854C]01             cmp     byte [bQueryOpt],1
44755 00004767 75F8                     jnz     short disable_exit
44756 00004769 F606[7B4C]02             test    byte [dae_flag],2
44757 0000476E 75F1                     jnz     short disable_exit
44758 00004770 800E[7B4C]02             or      byte [dae_flag],2
44759                                   ;mov    dx,' Y' ; (MASM syntax) ; 2059h
44760                                   ; 10/09/2023 (BugFix)
44761 00004775 BA5920                   mov     dx,'Y ' ; (NASM syntax) ; 2059h
44762 00004778 EBB9                     jmp     short dae_1
44763
44764                                   ;endif  ;MULTI_CONFIG
44765
44766                                   ;%endif     ; 02/11/2022
44767
44768
44769                                   ; 19/04/2019 - Retro DOS v4.0
44770
44771                                   ;--------------------------------------------------------------------------
44772                                   ;
44773                                   ; procedure : delim
44774                                   ;
44775                                   ;--------------------------------------------------------------------------
44776
44777                                   ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
44778                                   ; (SYSINIT:4C45h)
44779
44780                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
44781                                   ;%if 0
44782                                   ;;ifdef     MULTI_CONFIG
44783                                   ;
44784                                   any_delim:
44785 0000477A 3C0D                         cmp     al,cr
44786 0000477C 7427                         je      short delim_ret
44787 0000477E 3C0A                         cmp     al,lf
44788 00004780 7423                         je      short delim_ret
44789 00004782 3C5B                         cmp     al,'['
44790 00004784 741F                         je      short delim_ret
44791 00004786 3C5D                         cmp     al,']'
44792 00004788 741B                         je      short delim_ret
44793                                   ;
44794                                   ;;endif ;MULTI_CONFIG
44795                                   ;%endif ; 02/11/2022
44796
44797                                       ; 02/11/2022 - Retrodos v4.0 (Modified MSDOS 5.0 IO.SYS)
44798                                       ; (SYSINIT:3450h)
44799                                   delim:
44800 0000478A 3C2F                         cmp     al,'/'          ; ibm will assume "/" as an delimeter.
44801 0000478C 7417                         je      short delim_ret
44802
44803 0000478E 3C00                         cmp     al,0            ; special case for sysinit!!!
44804 00004790 7413                         je      short delim_ret
44805
44806                                   org_delim:                  ; used by organize routine except for getting
44807 00004792 3C20                         cmp     al,' '           ; the filename.
44808 00004794 740F                         je      short delim_ret
44809 00004796 3C09                             cmp     al,tab ; 9
44810 00004798 740B                         je      short delim_ret
44811 0000479A 3C3D                         cmp     al,'='
44812 0000479C 7407                         je      short delim_ret
44813 0000479E 3C2C                         cmp     al,','
44814 000047A0 7403                         je      short delim_ret
44815 000047A2 3C3B                         cmp     al,';'
44816
44817                                       ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
44818
44819                                   ; 04/01/2023 - Retro DOS v4.2
44820                                   ;ifdef      MULTI_CONFIG
44821                                   ;   Make sure there's no chance of a false EOF indication
44822 000047A4 F8                           clc
44823                                   ;endif
44824                                       ; 02/11/2022
44825                                   delim_ret:
44826                                       ; 04/01/2023
44827                                       ; cf = 0
44828                                   nl_ret:     ; 10/09/2023
44829 000047A5 C3                           retn
44830
44831                                   ;--------------------------------------------------------------------------
44832                                   ;
44833                                   ; procedure : newline
44834                                   ;
44835                                   ;  newline returns with first character of next line
44836                                   ;
44837                                   ;--------------------------------------------------------------------------
44838
44839                                   newline:
44840 000047A6 E80600                       call    getchr                  ;skip non-control characters
44841 000047A9 72FA                         jc      short nl_ret
44842 000047AB 3C0A                         cmp     al,lf                   ;look for line feed
44843 000047AD 75F7                         jne     short newline
44844
44845                                       ; 10/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
44846                                       ;call   getchr
44847                                   ;nl_ret:
44848                                       ;retn
44849                                       ; 10/09/2023
44850                                       ;jmp    short getchr
44851
44852                                   ; 10/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
```

```
                                %if 1

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : getchr
                                ;
                                ;--------------------------------------------------------------------------

                                    ; 24/10/2022
                                getchr:
                                    ; 12/12/2022
                                    ;push   cx
                                    ;mov    cx,[count]
                                    ;jcxz   nochar
                                    ; 12/12/2022
000047AF 833E[5603]01               cmp     word [count],1
000047B4 720F                       jb      short nochar ; cf=1 ([count] = 0)

000047B6 8B36[5A03]                 mov     si,[chrptr]
000047BA 268A04                     mov     al,[es:si]
000047BD FF0E[5603]                 dec     word [count]
000047C1 FF06[5A03]                 inc     word [chrptr]
                                    ; 12/12/202
                                    ; cf=0
                                    ;clc
                                ;get_ret:
                                    ;pop    cx
                                    ;retn
                                nochar:
                                    ; 12/12/2022
                                    ; cf=1
                                    ;stc
                                    ;jmp    short get_ret

000047C5 C3                         retn
                                %endif

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : mapcase
                                ;
                                ;--------------------------------------------------------------------------

                                    ; 02/11/2022 - Retro DOS 4.0 (Modified MSDOS 5.0 IO.SYS)

                                    ; 04/01/2023 - Retro DOS 4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:4C7Eh)
                                mapcase:
000047C6 51                         push    cx
000047C7 56                         push    si
000047C8 1E                         push    ds

000047C9 06                         push    es
000047CA 1F                         pop     ds

                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)

                                ; 04/01/2023 - Retro DOS 4.2

                                ;ifdef      MULTI_CONFIG
000047CB 88C3                       mov     bl,al                   ; same cmd code this line
                                ;else
                                ;   xor     si,si
                                ;endif
                                    ; 02/11/2022
                                    ; 04/01/2023 - Retro DOS 4.2
                                    ;xor    si, si

                                convloop:
000047CD AC                         lodsb
000047CE 3C61                       cmp     al,'a'
000047D0 7209                       jb      short noconv
000047D2 3C7A                       cmp     al,'z'
000047D4 7705                       ja      short noconv
000047D6 2C20                       sub     al,20h
000047D8 8844FF                     mov     [si-1],al
                                noconv:

                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)

                                ; 04/01/2023 - Retro DOS 4.2
                                ;ifdef      MULTI_CONFIG

                                ;   When MULTI_CONFIG enabled, "mapcase" is used to map everything to
                                ;   upper-case a line at a time, after we've been able to figure out whether
                                ;   the line is a SET command or not (since we don't want to upper-case
                                ;   anything after the "=" in a SET)
                                ;
000047DB 80FB56                     cmp     bl,CONFIG_SET  ; 'V'  ; preserve case for part of the line?
000047DE 7504                       jne     short check_eol             ; no, just check for end-of-line
000047E0 3C3D                       cmp     al,'='                ; separator between SET var and value?
000047E2 740A                       je      short convdone        ; yes
                                check_eol:
000047E4 3C0D                       cmp     al,cr
000047E6 7406                       je      short convdone
000047E8 3C0A                       cmp     al,lf
000047EA 7402                       je      short convdone
                                ;endif
                                    ; 02/11/2022
000047EC E2DF                       loop    convloop
                                convdone:
000047EE 1F                         pop     ds
000047EF 5E                         pop     si
000047F0 59                         pop     cx
000047F1 C3                         retn

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : round
                                ;
                                ; round the values in memlo and memhi to paragraph boundary.
                                ; perform bounds check.
                                ;
                                ;--------------------------------------------------------------------------

                                round:
                                    ; 10/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
000047F2 1E                         push    ds
000047F3 0E                         push    cs
000047F4 1F                         pop     ds

000047F5 50                         push    ax
                                    ;mov    ax,[cs:memlo]
000047F6 A1[6203]                   mov     ax,[memlo]
```

```
                                    call    ParaRound               ; para round up
                                    ;add    [cs:memhi],ax
000047FC 0106[6403]                 add     [memhi],ax
                                    ;mov    word [cs:memlo],0
00004800 C706[6203]0000             mov     word [memlo],0
                                    ;mov    ax,[cs:memhi]           ; ax = new memhi
00004806 A1[6403]                   mov     ax,[memhi]
                                    ;cmp    ax,[cs:ALLOCLIM]        ; if new memhi >= alloclim, error
00004809 3B06[A502]                 cmp     ax,[ALLOCLIM]
                                    ;jae    short mem_err
                                    ; 13/04/2024
0000480D 7322                       jae     short mem_err2 ; ds = cs
                                    ;test   byte [cs:setdevmarkflag],for_devmark ; 2
0000480F F606[6919]02               test    byte [setdevmarkflag],for_devmark ; 2
00004814 7416                       jz      short skip_set_devmarksize
00004816 06                         push    es
00004817 56                         push    si
                                    ;mov    si,[cs:devmark_addr]
00004818 8B36[6719]                 mov     si,[devmark_addr]
0000481C 8EC6                       mov     es,si
0000481E 29F0                       sub     ax,si
00004820 48                         dec     ax
                                    ;mov    [es:3],ax
00004821 26A30300                   mov     [es:devmark.size],ax  ; paragraph
                                    ;and    byte [cs:setdevmarkflag],not_for_devmark ; 0FDh
00004825 8026[6919]FD               and     byte [setdevmarkflag],not_for_devmark ; 0FDh
0000482A 5E                         pop     si
0000482B 07                         pop     es
                                skip_set_devmarksize:
0000482C 58                         pop     ax

                                    ; 10/09/2023
0000482D 1F                         pop     ds

                                    ; 11/12/2022
                                    ; cf = 0
                                    ; 02/11/2022
                                    ;clc    ; ? (not needed here) ; clear carry
0000482E C3                         retn

                                ;---------------------------------------------------------------------------

                                mem_err:
                                    ; 11/12/2022
0000482F 0E                         push    cs
00004830 1F                         pop     ds
                                mem_err2:
00004831 BA[4951]                   mov     dx,badmem
                                    ;push   cs
                                    ;pop    ds
00004834 E81D02                     call    print
00004837 E914CB                     jmp     stall

                                ;---------------------------------------------------------------------------
                                ;
                                ; procedure : calldev
                                ;
                                ;---------------------------------------------------------------------------

                                    ; 02/11/2022 - Retrodos v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:34E0h)

                                    ; 13/04/2024 - Retrodos v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:4F3Eh)

                                calldev:
0000483A 2E8E1E[3B24]               mov     ds,[cs:DevEntry+2]
0000483F 2E031E[3924]               add     bx,[cs:DevEntry]      ; do a little relocation
00004844 8B07                       mov     ax,[bx]

00004846 2EFF36[3924]               push    word [cs:DevEntry]
0000484B 2EA3[3924]                 mov     [cs:DevEntry],ax
0000484F BB[6F03]                   mov     bx,packet
00004852 2EFF1E[3924]               call    far [cs:DevEntry]
00004857 2E8F06[3924]               pop     word [cs:DevEntry]
0000485C C3                         retn

                                ;---------------------------------------------------------------------------
                                ;
                                ; procedure : todigit
                                ;
                                ;---------------------------------------------------------------------------

                                todigit:
0000485D 2C30                       sub     al,'0'
                                    ;jb     short notdig  ; 02/11/2022
                                    ; 12/12/2022
0000485F 7203                       jb      short notdig2
                                    ;cmp    al,9
                                    ;ja     short notdig
                                    ;clc
                                    ;retn
                                    ; 12/12/2022
00004861 3C0A                       cmp     al,10
00004863 F5                         cmc
                                notdig:
                                    ;stc
                                notdig2:
00004864 C3                         retn

                                ;---------------------------------------------------------------------------
                                ;
                                ; procedure : getnum
                                ;
                                ; getnum parses a decimal number.
                                ; returns it in ax, sets zero flag if ax = 0 (may be considered an
                                ; error), if number is bad carry is set, zero is set, ax=0.
                                ;
                                ;---------------------------------------------------------------------------

                                getnum:
00004865 53                         push    bx
00004866 31DB                       xor     bx,bx                   ; running count is zero
                                b2:
00004868 E8F2FF                     call    todigit                 ; do we have a digit ?
0000486B 7247                       jc      short badnum            ; no, bomb

0000486D 93                         xchg    ax,bx                   ; put total in ax
0000486E 53                         push    bx                      ; save digit (0 to 9)
                                    ;mov    bx,10                   ; base of arithmetic
                                    ; 12/12/2022
0000486F B30A                       mov     bl,10
```

```
45101 00004871 F7E3                     mul     bx                      ; shift by one decimal digit
45102 00004873 5B                       pop     bx                      ; get back digit (0 to 9)
45103 00004874 00D8                     add     al,bl                   ; get total
45104 00004876 80D400                   adc     ah,0                    ; make that 16 bits
45105 00004879 7239                     jc      short badnum            ; too big a number
45106
45107 0000487B 93                       xchg    ax,bx                   ; stash total
45108
45109 0000487C E830FF                   call    getchr                  ;get next digit
45110 0000487F 722D                     jc      short b1                ; no more characters
45111 00004881 3C20                     cmp     al,' '                  ; space?
45112 00004883 741F                     je      short b15               ; then end of digits
45113 00004885 3C2C                     cmp     al,','                  ; ',' is a seperator!!!
45114 00004887 741B                     je      short b15               ; then end of digits.
45115 00004889 3C09                     cmp     al, tab ; 9             ; tab
45116 0000488B 7417                     je      short b15
45117 0000488D 2E3A06[AE02]             cmp     al,[cs:sepchr]          ; allow 0 or special separators
45118 00004892 7410                     je      short b15
45119 00004894 3C2F                     cmp     al,'/'                  ; see if another switch follows
45120                                   ; 12/12/2022
45121                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
45122                                   ;nop                            ; cas - remnant of old bad code
45123                                   ;nop
45124 00004896 740C                     je      short b15
45125 00004898 3C0A                     cmp     al,lf                   ; line-feed?
45126 0000489A 7408                     je      short b15
45127 0000489C 3C0D                     cmp     al,cr                   ; carriage return?
45128 0000489E 7404                     je      short b15
45129 000048A0 08C0                     or      al,al                   ; end of line separator?
45130 000048A2 75C4                     jnz     short b2                ; no, try as a valid char...
45131                           b15:
45132 000048A4 2EFF06[5603]             inc     word [cs:count]                 ; one more character to s...
45133 000048A9 2EFF0E[5A03]             dec     word [cs:chrptr]        ; back up over separator
45134                           b1:
45135 000048AE 89D8                     mov     ax,bx                   ; get proper count
45136 000048B0 09C0                     or      ax,ax                   ; clears carry, sets zero accordingly
45137 000048B2 5B                       pop     bx
45138 000048B3 C3                       retn
45139                           badnum:
45140                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
45141                                   ;mov    byte [cs:sepchr],0
45142 000048B4 31C0                     xor     ax,ax                   ; set zero flag, and ax = 0
45143                                   ; 12 /12/2022
45144 000048B6 2EA2[AE02]               mov     [cs:sepchr],al ; 0
45145 000048BA 5B                       pop     bx
45146 000048BB F9                       stc                             ; and carry set
45147 000048BC C3                       retn
45148
45149                           ;******************************************************************************
45150
45151                           setdoscountryinfo:
45152
45153                           ;------------------------------------------------------------------------------
45154                           ;input: es:di -> pointer to dos_country_cdpg_info
45155                           ;   ds:0  -> buffer.
45156                           ;   si = 0
45157                           ;   ax = country id
45158                           ;   dx = code page id. (if 0, then use ccsyscodepage as a default.)
45159                           ;   bx = file handle
45160                           ;   this routine can handle maximum 438 country_data entries.
45161                           ;
45162                           ;output: dos_country_cdpg_info set.
45163                           ;    carry set if any file read failure or wrong information in the file.
45164                           ;    carry set and cx = -1 if cannot find the matching country_id,
45165                           ;    codepage_id in the file.
45166                           ;------------------------------------------------------------------------------
45167
45168                                   ; 04/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
45169                                   ; (SYSINIT:4D83h)
45170
45171                                   ; 10/09/2023 - Retrodos v4.2 IO.SYS (Optimization)
45172                                   ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:4FCAh)
45173
45174 000048BD 57                       push    di
45175 000048BE 50                       push    ax
45176 000048BF 52                       push    dx
45177
45178 000048C0 31C9                     xor     cx,cx
45179 000048C2 31D2                     xor     dx,dx
45180 000048C4 B80002                   mov     ax,512                  ;read 512 bytes
45181 000048C7 E84301                   call    readincontrolbuffer     ;read the file header
45182 000048CA 724A                     jc      short setdosdata_fail
45183
45184 000048CC 06                       push    es
45185 000048CD 56                       push    si
45186
45187 000048CE 0E                       push    cs
45188 000048CF 07                       pop     es
45189
45190 000048D0 BF[204B]                 mov     di,country_file_signature ; db 0FFh,'COUNTRY'
45191 000048D3 B90800                   mov     cx,8                    ;length of the signature
45192 000048D6 F3A6                     repz    cmpsb
45193
45194 000048D8 5E                       pop     si
45195 000048D9 07                       pop     es
45196 000048DA 753A                     jnz     short setdosdata_fail ;signature mismatch
45197
45198 000048DC 83C612                   add     si,18                   ;si -> county info type
45199 000048DF 803C01                   cmp     byte [si],1             ;only accept type 1 (currently only 1 header type)
45200 000048E2 7532                     jne     short setdosdata_fail ;cannot proceed. error return
45201
45202 000048E4 46                       inc     si                      ;si -> file offset
45203 000048E5 8B14                     mov     dx,[si]                 ;get the info file offset.
45204 000048E7 8B4C02                   mov     cx,[si+2]
45205 000048EA B80018                   mov     ax,6144                 ;read 6144 bytes.
45206 000048ED E81D01                   call    readincontrolbuffer     ;read info
45207 000048F0 7224                     jc      short setdosdata_fail
45208
45209 000048F2 8B0C                     mov     cx,[si]                 ;get the # of country, codepage combination entries
45210 000048F4 81F9B601                 cmp     cx,438                  ;cannot handle more than 438 entries.
45211 000048F8 771C                     ja      short setdosdata_fail
45212
45213 000048FA 46                       inc     si
45214 000048FB 46                       inc     si                      ;si -> entry information packet
45215 000048FC 5A                       pop     dx                      ;restore code page id
45216 000048FD 58                       pop     ax                      ;restore country id
45217 000048FE 5F                       pop     di
45218
45219                           setdoscntry_find:               ;search for desired country_id,codepage_id.
45220 000048FF 3B4402                   cmp     ax,[si+2]               ;compare country_id
45221 00004902 7509                     jne     short setdoscntry_next
45222
45223                                   ;cmp    dx,0                    ;no user specified code page ?
45224                                   ;je     short setdoscntry_any_codepage ;then no need to match code page id.
```

```
                                    ; 10/09/2023
00004904 09D2                       or      dx,dx ; cmp dx,0
00004906 7413                       jz      short setdoscntry_any_codepage
00004908 3B5404                     cmp     dx,[si+4]               ;compare code page id
0000490B 7411                       je      short setdoscntry_got_it

                                setdoscntry_next:
0000490D 0334                       add     si,[si]                 ;next entry
0000490F 46                         inc     si
00004910 46                         inc     si                      ;take a word for size of entry itself
00004911 E2EC                       loop    setdoscntry_find

                                    ;mov    cx,-1                   ;signals that bad country id entered.
                                    ; 10/09/2023
00004913 49                         dec     cx ; 0 -> -1
                                setdoscntry_fail:
00004914 F9                         stc
00004915 C3                         retn

                                setdosdata_fail:
00004916 5E                         pop     si
00004917 59                         pop     cx
00004918 5F                         pop     di
00004919 EBF9                       jmp     short setdoscntry_fail

                                setdoscntry_any_codepage:           ;use the code_page_id of the country_id found.
0000491B 8B5404                     mov     dx,[si+4]

                                setdoscntry_got_it:                         ;found the matching entry
0000491E 2E8916[284B]               mov     [cs:cntrycodepage_id],dx ;save code page id for this country.
00004923 8B540A                     mov     dx,[si+10]              ;get the file offset of country data
00004926 8B4C0C                     mov     cx,[si+12]
00004929 B80002                     mov     ax,512                  ;read 512 bytes
0000492C E8DE00                     call    readincontrolbuffer
0000492F 72E3                       jc      short setdoscntry_fail

00004931 8B0C                       mov     cx,[si]                 ;get the number of entries to handle.
00004933 46                         inc     si
00004934 46                         inc     si                      ;si -> first entry

                                setdoscntry_data:
00004935 57                         push    di                      ;es:di -> dos_country_cdpg_info
00004936 51                         push    cx                      ;save # of entry left
00004937 56                         push    si                      ;si -> current entry in control buffer

00004938 8A4402                     mov     al,[si+2]               ;get data entry id
0000493B E8A400                     call    getcountrydestination ;get the address of destination in es:di
0000493E 727C                       jc      short setdoscntry_data_next ;no matching data entry id in dos

00004940 8B5404                     mov     dx,[si+4]               ;get offset of data
00004943 8B4C06                     mov     cx,[si+6]
00004946 B80042                     mov     ax,4200h
00004949 F9                         stc
0000494A CD21                       int     21h                     ;move pointer
0000494C 72C8                       jc      short setdosdata_fail

0000494E BA0002                     mov     dx,512                  ;start of data buffer
00004951 B91400                     mov     cx,20                   ;read 20 bytes only. we only need to
00004954 B43F                       mov     ah,3Fh                  ;look at the length of the data in the file.
00004956 F9                         stc
00004957 CD21                       int     21h                     ;read the country.sys data
00004959 72BB                       jc      short setdosdata_fail ;read failure

0000495B 39C8                       cmp     ax,cx
0000495D 75B7                       jne     short setdosdata_fail ; 13/05/2019

0000495F 8B5404                     mov     dx,[si+4]               ;get offset of data again.
00004962 8B4C06                     mov     cx,[si+6]
00004965 B80042                     mov     ax,4200h
00004968 F9                         stc
00004969 CD21                       int     21h                     ;move pointer back again
0000496B 72A9                       jc      short setdosdata_fail

0000496D 56                         push    si
0000496E BE0802                     mov     si,(512+8)              ;get length of the data from the file
00004971 8B0C                       mov     cx,[si]
00004973 5E                         pop     si
00004974 BA0002                     mov     dx,512                  ;start of data buffer
00004977 83C10A                     add     cx,10                   ;signature + a word for the length itself
0000497A B43F                       mov     ah,3Fh                  ;read the data from the file.
0000497C F9                         stc
0000497D CD21                       int     21h
0000497F 7295                       jc      short setdosdata_fail

00004981 39C8                       cmp     ax,cx
00004983 7591                       jne     short setdosdata_fail

00004985 8A4402                     mov     al,[si+2]               ;save data id for future use.
00004988 BE0802                     mov     si,(512+8)              ;si-> data buffer + id tag field
0000498B 8B0C                       mov     cx,[si]                 ;get the length of the file
0000498D 41                         inc     cx                      ;take care of a word for lenght of tab
0000498E 41                         inc     cx                      ;itself.
0000498F 81F9F805                   cmp     cx,(2048-512-8)         ; 1528 ;fit into the buffer?
00004993 7781                       ja      short setdosdata_fail

                                    ;if     bugfix
00004995 E83100                     call    setdbcs_before_copy
                                    ;endif

00004998 3C01                       cmp     al,SetCountryInfo ; 1 ;is the data for setcountryinfo table?
0000499A 7511                       jne     short setdoscntry_mov ;no, don't worry

0000499C 26FF7518                   push    word [es:di+country_cdpg_info.ccMono_Ptr-country_cdpg_info.ccCountryInfoLen]
                                    ;push   word [es:di+24]                 ;cannot destroy ccmono_ptr address. save them.
000049A0 26FF751A                   push    word [es:di+country_cdpg_info.ccMono_Ptr-country_cdpg_info.ccCountryInfoLen+2]
                                    ;push   word [es:di+26]                 ;at this time di -> cccountryinfolen

000049A4 57                         push    di                      ;save di

                                    ;push   ax
                                    ;mov    ax,[cs:cntrycodepage_id] ;do not use the code page info in country_info
                                    ;mov    [si+4],ax               ;use the saved one for this !!!!
                                    ;pop    ax
                                    ; 10/09/2023
000049A5 2EFF36[284B]               push    word [cs:cntrycodepage_id]
000049AA 8F4404                     pop     word [si+4]

                                setdoscntry_mov:
000049AD F3A4                       rep     movsb                   ;copy the table into dos
000049AF 3C01                       cmp     al,SetCountryInfo       ;was the ccmono_ptr saved?
000049B1 7509                       jne     short setdoscntry_data_next

000049B3 5F                         pop     di                      ;restore di
000049B4 268F451A                   pop     word [es:di+country_cdpg_info.ccMono_Ptr-country_cdpg_info.ccCountryInfoLen+2]
```

```
                                    ;pop    word [es:di+26]             ;restore
000049B8 268F4518                   pop     word [es:di+country_cdpg_info.ccMono_Ptr-country_cdpg_info.ccCountryInfoLen]
                                    ;pop    word [es:di+24]

                                setdoscntry_data_next:
000049BC 5E                         pop     si                  ;restore control buffer pointer
000049BD 59                         pop     cx                  ;restore # of entries left
000049BE 5F                         pop     di                  ;restore pointer to dso_country_cdpg
000049BF 0334                       add     si,[si]             ;try to get the next entry
000049C1 46                         inc     si
000049C2 46                         inc     si                  ;take a word of entry length itself
000049C3 49                         dec     cx
                                    ; 10/09/2023
000049C4 741B                       jz      short setdoscntry_ok
                                    ;cmp    cx,0
                                    ;je     short setdoscntry_ok
000049C6 E96CFF                     jmp     setdoscntry_data

                                    ; 18/12/2022
                                ;setdoscntry_ok:
                                    ;retn

                                ;--------------------------------------------------------------------------

                                    ;if     bugfix

                                    ; 10/09/2023 - Retrodos v4.2 IO.SYS (Optimization)

                                setdbcs_before_copy:
000049C9 3C07                       cmp     al,SetDBCS ; 7      ; dbcs vector set?
000049CB 7514                       jne     short sdbcsbc       ; jump if not

                                    ; 10/09/2023
000049CD 50                         push    ax
000049CE 31C0                       xor     ax,ax
000049D0 263905                     cmp     [es:di],ax ; 0
000049D3 740B                       je      short sdbcsbc_pop

                                    ;cmp    word [es:di],0      ; zero byte data block?
                                    ;je     short sdbcsbc       ; jump if so

000049D5 57                         push    di
                                    ; 10/09/2023
                                    ;push   ax
000049D6 51                         push    cx
000049D7 268B0D                     mov     cx,[es:di]          ; load block length
                                    ;add    di,2                ; points actual data
000049DA 47                         inc     di
000049DB 47                         inc     di
                                    ;xor    al,al               ; fill bytes
000049DC F3AA                       rep     stosb               ; clear data block
000049DE 59                         pop     cx
                                    ;pop    ax
000049DF 5F                         pop     di

                                sdbcsbc_pop:        ; 10/09/2023
000049E0 58                         pop     ax
                                sdbcsbc:
                                setdoscntry_ok:     ; 18/12/2022
000049E1 C3                         retn

                                    ;endif

                                ;--------------------------------------------------------------------------

                                getcountrydestination:

                                ;--------------------------------------------------------------------------
                                ;get the destination address in the dos country info table.
                                ;
                                ;input: al - data id
                                ;   es:di -> dos_country_cdpg_info
                                ;on return:
                                ;   es:di -> destination address of the matching data id
                                ;   carry set if no matching data id found in dos.
                                ;--------------------------------------------------------------------------

                                    ; 04/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:4EB2h)

                                    ; 10/09/2023 - Retrodos v4.2 IO.SYS (Optimization)
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:50F9h)

000049E2 51                         push    cx
                                    ;add    di,74
000049E3 83C74A                     add     di,country_cdpg_info.ccNumber_of_entries
                                                                    ;skip the reserved area, syscodepage etc.
000049E6 268B0D                     mov     cx,[es:di]              ;get the number of entries
000049E9 47                         inc     di
000049EA 47                         inc     di                      ;si -> the first start entry id

                                getcntrydest:
000049EB 263805                     cmp     byte [es:di],al
000049EE 7413                       je      short getcntrydest_ok

000049F0 26803D01                   cmp     byte [es:di],SetCountryInfo ;was it setcountryinfo entry?
000049F4 7405                       je      short getcntrydest_1

000049F6 83C705                     add     di,5                    ;next data id
000049F9 EB03                       jmp     short getcntrydest_loop

                                getcntrydest_1:
                                    ;add    di,41
000049FB 83C729                     add     di,NEW_COUNTRY_SIZE+3 ;next data id
                                getcntrydest_loop:
000049FE E2EB                       loop    getcntrydest
00004A00 F9                         stc
                                    ;jmp    short getcntrydest_exit
                                getcntrydest_exit:
                                    ; 10/09/2023
00004A01 59                         pop     cx
00004A02 C3                         retn

                                getcntrydest_ok:
                                    ; 10/09/2023
00004A03 47                         inc     di

                                ;   cmp     al,SetCountryInfo ; 1 ;select country info?
                                ;   jne     short getcntrydest_ok1
                                ;
                                ;   ;inc    di                      ;now di -> cccountryinfolen
                                ;   jmp     short getcntrydest_exit

                                    ; 10/09/2023
```

```
00004A04 3C01                       cmp     al,SetCountryInfo ; 1 ;select country info?
00004A06 74F9                       je      short getcntrydest_exit

                                getcntrydest_ok1:
                                    ;les    di,[es:di+1]          ;get the destination in es:di
                                    ; 10/09/2023
00004A08 26C43D                     les     di,[es:di]
                                ;getcntrydest_exit:
00004A0B 59                         pop     cx
00004A0C C3                         retn

                                ;--------------------------------------------------------------------------

                                readincontrolbuffer:

                                ;--------------------------------------------------------------------------
                                ;move file pointer to cx:dx
                                ;read ax bytes into the control buffer. (should be less than 2 kb)
                                ;si will be set to 0 hence ds:si points to the control buffer.
                                ;
                                ;entry:  cx,dx offset from the start of the file where the read/write pointer
                                ;    be moved.
                                ;    ax - # of bytes to read
                                ;    bx - file handle
                                ;    ds - buffer seg.
                                ;return: the control data information is read into ds:0 - ds:0200.
                                ;    cx,dx value destroyed.
                                ;    carry set if error in reading file.
                                ;--------------------------------------------------------------------------

00004A0D 50                         push    ax                  ;# of bytes to read
00004A0E B80042                     mov     ax,4200h
00004A11 F9                         stc
00004A12 CD21                       int     21h                 ;move pointer
00004A14 59                         pop     cx                  ;# of bytes to read
00004A15 7209                       jc      short ricb_exit

00004A17 31D2                       xor     dx,dx               ;ds:dx -> control buffer
00004A19 31F6                       xor     si,si
00004A1B B43F                       mov     ah,3Fh              ;read into the buffer
00004A1D F9                         stc
00004A1E CD21                       int     21h                 ;should be less than 1024 bytes.
                                ricb_exit:
00004A20 C3                         retn

                                ;--------------------------------------------------------------------------

                                ;! set_country_path procedure is not called from anywhere !
                                ; Erdogan Tan - 04/08/2023
                                %if 0

                                set_country_path:

                                ;--------------------------------------------------------------------------
                                ;in:  ds - sysinitseg, es - confbot, si -> start of the asciiz path string
                                ;     dosinfo_ext, cntry_drv, cntry_root, cntry_path
                                ;     assumes current directory is the root directory.
                                ;out: ds:di -> full path (cntry_drv).
                                ;     set the cntry_drv string from the country=,,path command.
                                ;     ds, es, si value saved.
                                ;--------------------------------------------------------------------------

                                    ; 04/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:4EF4h)

                                    ; 10/09/2023 - Retrodos v4.2 IO.SYS (Optimization)
                                    ; (Retrodos v5.0 Pre-Works)
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:513Bh)

                                    push    si

                                    push    ds                  ;switch ds, es
                                    push    es
                                    pop     ds
                                    pop     es                  ;now ds -> confbot, es -> sysinitseg

                                    call    chk_drive_letter    ;current ds:[si] is a drive letter?
                                    jc      short scp_default_drv ;no, use current default drive.

                                    mov     al,[si]
                                    inc     si
                                    inc     si                  ;si -> next char after ":"
                                    jmp     short scp_setdrv

                                scp_default_drv:
                                    mov     ah,19h
                                    int     21h
                                    add     al,"A"              ;convert it to a character.

                                scp_setdrv:
                                    mov     [cs:cntry_drv],al   ;set the drive letter.
                                    mov     di,cntry_path
                                    mov     al,[si]
                                    cmp     al, "\"
                                    je      short scp_root_dir

                                    cmp     al,"/"              ;let's accept "/" as an directory delim
                                    ;je     short scp_root_dir
                                    ;jmp    short scp_path
                                    ; 04/01/2023
                                    jne     short scp_path

                                scp_root_dir:
                                    dec     di                  ;di -> cntry_root
                                scp_path:
                                    call    move_asciiz         ;copy it

                                    mov     di,cntry_drv
                                scpath_exit:

                                    push    ds                  ;switch ds, es
                                    push    es
                                    pop     ds
                                    pop     es                  ;ds, es value restored

                                    pop     si
                                    retn

                                ;--------------------------------------------------------------------------

                                chk_drive_letter:

                                ;check if ds:[si] is a drive letter followed by ":".
                                ;assume that every alpha character is already converted to upper case.
```

```
                        ;carry set if not.
                        
                            ; 04/01/2023 - Retrodos v4.2
                        
                            push    ax
                            cmp     byte [si],"A"
                            ;jb     short cdletter_no
                            jb      short cdletter_exit
                            cmp     byte [si],"Z"
                            ja      short cdletter_no
                            cmp     byte [si+1],":"
                            ;jne    short cdletter_no
                            ;jmp    short cdletter_exit
                            ; 04/01/2023
                            je      short cdletter_exit
                        
                        cdletter_no:
                            stc
                        cdletter_exit:
                            pop     ax
                            retn
                        
                        %endif
                        
                        ;--------------------------------------------------------------------------
                        
                        move_asciiz:
                        
                        ;in: ds:si -> source es:di -> target
                        ;out: copy the string until 0.
                        ;assumes there exists a 0.
                        
                            ; 10/09/2023 - Retrodos v4.2 IO.SYS (Optimization)
                            ; (MSDOS 6.21 IO.SYS - SYSINIT:4F40h)
                            ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:5187h)
                        
                        masciiz_loop:
                            ; 10/09/2023
00004A21 F604FF             test    byte [si],0FFh
00004A24 A4                 movsb
                            ;cmp    byte [si-1],0  ; was it 0?
                            ;jne    short masciiz_loop
00004A25 75FA               jnz     short masciiz_loop ; 10/09/2023
00004A27 C3                 retn
                        
                        ;--------------------------------------------------------------------------
                        
                        ;   ds:dx points to string to output (asciz)
                        ;
                        ;   prints <badld_pre> <string> <badld_post>
                        
                        badfil:
00004A28 0E                 push    cs
00004A29 07                 pop     es
                        
00004A2A 89D6               mov     si,dx
                        badload:
00004A2C BA[B350]           mov     dx,badld_pre   ; want to print config error
00004A2F BB[7050]           mov     bx,crlfm
                        prnerr:
00004A32 0E                 push    cs
00004A33 1F                 pop     ds ; *
00004A34 E81D00             call    print
                        prn1:
00004A37 268A14             mov     dl,[es:si]
00004A3A 08D2               or      dl,dl
00004A3C 7407               jz      short prn2
00004A3E B402               mov     ah,STD_CON_OUTPUT ; 2
00004A40 CD21               int     21h
00004A42 46                 inc     si
00004A43 EBF2               jmp     short prn1
                        prn2:
00004A45 89DA               mov     dx,bx
00004A47 E80A00             call    print
                            ; 11/12/2022
                            ; ds = cs ; *
00004A4A 803E[5503]01       cmp     byte [donotshownum],1
                                                    ; suppress line number when handling command.com
                            ;cmp    byte [cs:donotshownum],1
00004A4F 7407               je      short prnexit
                        
                            ; 18/12/2022
                            ;call   error_line
00004A51 E966E4             jmp     error_line
                        ;prnexit:
                            ;retn
                        
                        ;--------------------------------------------------------------------------
                        
                        print:
00004A54 B409               mov     ah,STD_CON_STRING_OUTPUT ; 9
00004A56 CD21               int     21h
                        prnexit:    ; 18/12/2022
00004A58 C3                 retn
                        
                        ;--------------------------------------------------------------------------
                        
                        ;  open device pointed to by dx, al has access code
                        ;   if unable to open do a device open null device instead
                        
                            ; 02/11/2022 - Retrodos v4.0 (Modified MSDOS 5.0 IO.SYS)
                            ; (SYSINIT:3764h)
                        open_dev:
00004A59 E80500             call    open_file
00004A5C 7309               jnc     short open_dev3
                        
                        open_dev1:
00004A5E BA[C14A]           mov     dx,nuldev
                            ; 18/12/2022
                            ;call   open_file
                        ;of_retn:
                            ;retn
                            ; 18/12/2022
                            ;jmp    short open_file
                        open_file:
00004A61 B43D               mov     ah,OPEN ; 3Dh
00004A63 F9                 stc
00004A64 CD21               int     21h
                        of_retn:    ; 18/12/2022
00004A66 C3                 retn
                        
                        open_dev3:
00004A67 89C3               mov     bx,ax               ; handle from open to bx
                            ;;xor   ax,ax               ; get device info
```

```
45721                                   ;;mov   ah,IOCTL ; 44h
45722                                   ;mov    ax,(IOCTL<<8) ; 13/05/2019
45723                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
45724                                   ;xor    ax,ax
45725                                   ;mov    ah,44h  ; IOCTL
45726                                   ; 11/12/2022
45727 00004A69 B80044                   mov     ax,4400h ; IOCTL<<8
45728
45729 00004A6C CD21                     int     21h
45730
45731 00004A6E F6C280                   test    dl,10000000b ; 80h
45732 00004A71 75F3                     jnz     short of_retn
45733
45734 00004A73 B43E                     mov     ah,CLOSE ; 3Eh
45735 00004A75 CD21                     int     21h
45736 00004A77 EBE5                     jmp     short open_dev1
45737
45738                                 ;--------------------------------------------------------------------------
45739
45740                                 ; 18/12/2022
45741                                 %if 0
45742                                 open_file:
45743                                     mov     ah,OPEN ; 3Dh
45744                                     stc
45745                                     int     21h
45746                                     retn
45747                                 %endif
45748
45749                                 ;--------------------------------------------------------------------------
45750
45751                                 ; test int24. return back to dos with the fake user response of "fail"
45752
45753                                 int24:
45754 00004A79 B003                      mov     al,3                    ; fail the system call
45755 00004A7B CF                        iret                            ; return back to dos.
45756
45757                                 ; 19/04/2019 - Retro DOS v4.0
45758
45759                                 ;--------------------------------------------------------------------------
45760                                 ; DATA
45761                                 ;--------------------------------------------------------------------------
45762
45763                                 ;include copyrigh.inc                   ; copyright statement
45764
45765                                 ; MSDOS 6.21 IO.SYS - SYSINIT:4FA3h
45766
45767                                 ;MsDosVersion6Copyr:
45768                                 ;   db      'MS DOS Version 6 (C)Copyright 1981-1993 Microsoft Corp '
45769                                 ;   db      'Licensed Material - Property of Microsoft All rights reserved '
45770
45771                                 ; 22/10/2022
45772                                 ; MSDOS 5.0 IO.SYS - SYSINIT:378Ch
45773
45774                                 ; 28/12/2022
45775                                 %if 0
45776                                     ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
45777                                 MsDosVersion5Copyr:
45778                                     db      'MS DOS Version 5.00 (C)Copyright 1981-1991 Microsoft Corp '
45779                                     db      'Licensed Material - Property of Microsoft All rights reserved '
45780                                 %endif
45781
45782                                 ; 13/04/2024 - Retro DOS v5.0
45783                                 ; PCDOS 7.1 IBMBIO.COM - SYSINIT:51EAh   (IBMBIO.COM offset 42266)
45784                                 %if 0
45785                                 IBMDOSV71COPYR:
45786                                     db 'IBM DOS Version 7.1 (C)Copyright 1981-2002 IBM Corporation '
45787                                     db 'Licensed Material - Property of IBM All rights reserved '
45788                                 %endif
45789
45790                                 ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
45791                                 ; 22/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
45792                                 ; 20/04/2019 - Retro DOS v4.0
45793                                 ;BOOTMES:
45794                                 ;   db      13
45795                                 ;   db      10
45796                                 ;   db      "MS-DOS version "
45797                                 ;   db      MAJOR_VERSION + "0"
45798                                 ;   db      "."
45799                                 ;   db      (MINOR_VERSION / 10) + "0"
45800                                 ;   db      (MINOR_VERSION % 10) + "0"
45801                                 ;   db      13,10
45802                                 ;   ;db    "Copyright 1981-1993 Microsoft Corp.",13,10,"$"
45803                                 ;   ; 22/10/2022
45804                                 ;   db      "Copyright 1981-1991 Microsoft Corp.",13,10,"$"
45805                                 ;   ;
45806                                 ;   db      0
45807
45808                                     ; 01/01/2023 - Retro DOS v4.2
45809
45810                                     ; 28/12/2022 - Retro DOS v4.1
45811                                 ;MsDosVersion5Copyr:
45812                                 ;   db      13,10,"MS DOS Version 5.0"
45813                                 ;   db      13,10,"Copyright 1981-1991 Microsoft Corp.",13,10,"$",0
45814
45815                                     ; 12/12/2022
45816 00004A7C 00                       db      0
45817                                 ; 12/12/2022
45818                                 BOOTMES:
45819 00004A7D 0D0A                     db      13,10
45820                                   ;;;db   "Retro DOS v4.0 (Modified MSDOS 5.0) "
45821                                   ; 28/12/2022
45822                                   ;;db    "Retro DOS v4.1 (Modified MSDOS 5.0) "
45823                                   ; 01/01/2023
45824                                   ;db     "Retro DOS v4.2 (Modified MSDOS 6.22) "
45825                                   ; 30/12/2023
45826 00004A7F 526574726F20444F53-      db      "Retro DOS v5.0 (Modified PCDOS 7.1) "
45826 00004A88 2076352E3020284D6F-
45826 00004A91 646966696564205043-
45826 00004A9A 444F5320372E312920
45827
45828 00004AA3 0D0A                     db      13,10
45829 00004AA5 6279204572646F6761-      db      "by Erdogan Tan [2024] " ; 01/01/2024
45829 00004AAE 6E2054616E205B3230-
45829 00004AB7 32345D20
45830 00004ABB 0D0A                     db      13,10
45831 00004ABD 0D0A2400                 db      13,10,"$",0
45832
45833 00004AC1 4E554C00             nuldev:     db      "NUL",0
45834 00004AC5 434F4E00             condev:     db      "CON",0
45835 00004AC9 41555800             auxdev:     db      "AUX",0
45836 00004ACD 50524E00             prndev:     db      "PRN",0
45837
45838                               ;IFDEF      CONFIGPROC
45839 00004AD1 5C434F4E4649472E53-  config:     db      "\CONFIG.SYS",0
```

```
00004ADA 595300
                                
00004ADD 413A                   cntry_drv:  db     "A:"
00004ADF 5C                     cntry_root: db     "\"
00004AE0 434F554E5452592E53-    cntry_path: db     "COUNTRY.SYS",0
00004AE9 595300
                                        ;db 52 dup (0)
00004AEC 00<rep 34h>                    times 52 db 0
                                
                                country_file_signature:
00004B20 FF434F554E545259           db     0FFh,'COUNTRY'
                                
                                cntrycodepage_id:
00004B28 0000                       dw     0
                                
                                ;ENDIF ; CONFIGPROC
                                
                                ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:5081h)
                                
                                ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;ifdef      MULTI_CONFIG
00004B2A 00                     newcmd:  db 0                       ; non-zero if non-std shell specified
00004B2B 40                     tmplate: db 64                        ; must precede commnd
                                ;endif
                                
                                ;ifdef ROMEXEC
                                ;   db      7                         ; size of commnd line (excl. null)
                                ;commnd: db "COMMAND",0
                                ;   db      56 dup (0)
                                ;else
                                    ; 02/11/2022
00004B2C 0C                         db      12                        ; size of commnd line (excl. null)
00004B2D 5C434F4D4D414E442E-     commnd:     db      "\COMMAND.COM",0
00004B36 434F4D00
                                    ;db     51 dup (0)
00004B3A 00<rep 33h>                times   51 db 0
                                ;endif
                                
                                ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;ifdef      MULTI_CONFIG
00004B6D 5C434F4D4D414E442E-    commnd2: db         "\COMMAND.COM",0      ; alternate commands to exec,
00004B76 434F4D00
00004B7A 022F5000                    db     2,"/P",0               ; followed by their respective alternate
00004B7E 5C4D53444F535C434F-    commnd3: db "\MSDOS\COMMAND.COM",0; command lines
00004B87 4D4D414E442E434F4D-
00004B90 00
00004B91 0B413A5C4D53444F53-         db     11,"A:\MSDOS /P",0    ;(the drive letter are dynamically replaced)
00004B9A 202F5000
00004B9E 5C444F535C434F4D4D-    commnd4: db "\DOS\COMMAND.COM",0  ;
00004BA7 414E442E434F4D00
00004BAF 09413A5C444F53202F-         db     9,"A:\DOS /P",0               ;
00004BB8 5000
                                def_swchr:
00004BBA 00                          db     0                      ; default switchchar (referenced as command_line-1)
                                ;endif
                                    ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                command_line:
00004BBB 022F50                     db      2,"/P"                ; default command.com args
                                    ;db     125 dup (0)
00004BBE 00<rep 7Dh>                times   125 db 0
                                
                                pathstring:
                                    ;db     64 dup (0)
00004C3B 00<rep 40h>                times   64 db 0
                                
                                ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:51D3h)
                                ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                
                                dae_flag:
00004C7B 00                         db      0 ; MSDOS 6.21 IO.SYS - SYSINIT:51D2h
                                
                                ;ifdef      MULTI_CONFIG
                                
                                ; 04/03/2022- Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS, SYSINIT)
                                MAX_MULTI_CONFIG equ 9     ; max # of multi-config menu items supported
                                
                                ;   Beware of byte pairs accessed as words (see all "KEEP AFTER" notes below)
                                
00004C7C 07                     bMenuColor: db      07h ; 1Fh       ; default fgnd/bgnd color
00004C7D 00                     bMenuPage:  db      0               ; menu video page (KEEP AFTER bMenuColor)
00004C7E 05                                 db      5               ; video page function # (KEEP AFTER bMenuPage)
00004C7F 00                     bLastCol:   db      0               ; ending column on status line
00004C80 18                     bLastRow:   db      24              ; row # of status line (KEEP AFTER bLastCol)
00004C81 00                     bDisableUI: db      0               ; 1=disable clean/interactive
                                                                      ; 2=disable default 2-second delay
00004C82 00                     bCRTPage:   db      0               ; value saved from BIOS data area
00004C83 0000                   wCRTStart:  dw      0               ; value saved from BIOS data area
00004C85 00                     bQueryOpt:  db      0               ; 0=off, 1=prompt all, 2=prompt none, 4=skip all
00004C86 01                     bDefBlock:  db      1               ; default block #
00004C87 00                     bMaxBlock:  db      0               ; maxmimum block #
00004C88 0000                   offDefBlock:        dw      0               ; offset of name of default block (if any)
00004C8A FF                     secTimeOut: db      -1 ; 0FFh       ; # of seconds for timeout (-1 == indefinite)
00004C8B 00                     secElapsed: db      0               ; # of seconds elapsed so far (KEEP AFTER secTimeOut)
00004C8C 00<rep Ah>             abBlockType:        times MAX_MULTI_CONFIG+1 db 0 ; array of block types
00004C96 0000<rep Ah>           aoffBlockName:      times MAX_MULTI_CONFIG+1 dw 0 ; array of offsets of block names
00004CAA 0000<rep Ah>           aoffBlockDesc:      times MAX_MULTI_CONFIG+1 dw 0 ; array of offsets of block descriptions
                                
00004CBE 434F4E4649473D00       szBoot:             db      "CONFIG=",0
00004CC6 4D454E5500             szMenu:             db      "MENU",0
00004CCB 434F4D4D4F4E00         szCommon:   db      "COMMON",0
                                
                                ;endif      ;MULTI_CONFIG
                                
                                    ; 10/09/2023
                                    ; MSDOS 6.21 IO.SYS - SYSINIT:5229h
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:546Eh)
                                
                                comtab:     ; label byte
                                
                                ;           cmd len    command         cmd code
                                ;           -------    -------         --------
                                
                                ;ifdef MULTI_CONFIG
00004CD2 015B5B                         db      1,      "[",            CONFIG_BEGIN
                                ;endif
00004CD5 05425245414B43                 db      5,      "BREAK",        CONFIG_BREAK
00004CDC 07425546464525342              db      7,      "BUFFERS",      CONFIG_BUFFERS
00004CE5 07434F4D4D454E5459             db      7,      "COMMENT",      CONFIG_COMMENT
00004CEE 07434F554E54525951             db      7,      "COUNTRY",      CONFIG_COUNTRY
00004CF7 0644455649434544               db      6,      "DEVICE",       CONFIG_DEVICE
00004CFF 0A4445564943454849-            db      10,     "DEVICEHIGH",   CONFIG_DEVICEHIGH
```

```
45954 00004D08 474855
45955 00004D0B 03444F5348                       db      3,      "DOS",          CONFIG_DOS
45956 00004D10 0845249565041524D-               db      8,      "DRIVPARM",     CONFIG_DRIVPARM
45956 00004D19 50
45957 00004D1A 044643425358                     db      4,      "FCBS",         CONFIG_FCBS
45958 00004D20 0546494C455346                   db      5,      "FILES",        CONFIG_FILES
45959                                   ;ifdef MULTI_CONFIG
45960 00004D27 07494E434C5544454A               db      7,      "INCLUDE",      CONFIG_INCLUDE
45961                                   ;endif
45962 00004D30 07494E5354414C4C49               db      7,      "INSTALL",      CONFIG_INSTALL
45963 00004D39 0B494E5354414C4C48-              db      11,     "INSTALLHIGH",  CONFIG_INSTALLHIGH
45963 00004D42 49474857
45964 00004D46 094C41535444524956-              db      9,      "LASTDRIVE",    CONFIG_LASTDRIVE
45964 00004D4F 454C
45965                                   ;ifdef MULTI_CONFIG
45966 00004D51 075355424D454E554F               db      7,      "SUBMENU",      CONFIG_SUBMENU
45967 00004D5A 094D454E55434F4C4F-              db      9,      "MENUCOLOR",    CONFIG_MENUCOLOR
45967 00004D63 5252
45968 00004D65 0B4D454E5544454641-              db      11,     "MENUDEFAULT",  CONFIG_MENUDEFAULT
45968 00004D6E 554C5441
45969 00004D72 084D454E554954454D-              db      8,      "MENUITEM",     CONFIG_MENUITEM
45969 00004D7B 45
45970                                   ;endif
45971 00004D7C 0A4D554C5449545241-              db      10,     "MULTITRACK",   CONFIG_MULTITRACK
45971 00004D85 434B4D
45972                                   ;ifdef MULTI_CONFIG
45973 00004D88 074E554D4C4F434B4E               db      7,      "NUMLOCK",      CONFIG_NUMLOCK
45974                                   ;endif
45975 00004D91 0352454D30                       db      3,      "REM",          CONFIG_REM
45976                                   ;ifdef MULTI_CONFIG
45977 00004D96 0353455456                       db      3,      "SET",          CONFIG_SET
45978                                   ;endif
45979 00004D9B 055348454C4C53                   db      5,      "SHELL",        CONFIG_SHELL
45980                                   ;if    STACKSW
45981 00004DA2 06535441434B534B                 db      6,      "STACKS",       CONFIG_STACKS
45982                                   ;endif
45983 00004DAA 085357495443484553-              db      8,      "SWITCHES",     CONFIG_SWITCHES
45983 00004DB3 31
45984                                       ; 18/03/2025 (BugFix)
45985                                       ;db     0
45986
45987                                       ; 10/09/2023
45988                                   ;aDosdata:  ; PCDOS 7.1 IBMBIO.COM - SYSINIT:5550h
45989                                       ; 13/04/2024 - Retro DOS v5.0
45990 00004DB4 07444F534441544154           db      7,      "DOSDATA",      CONFIG_DOSDATA ; 'T'
45991 00004DBD 00                           db      0
45992
45993                                   ;%endif ; 02/11/2022
45994
45995                                   ; 01/01/2023 - Retro DOS v4.2
45996                                   %if 0
45997
45998                                   comtab:
45999                                       ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
46000                                       ; (SYSINIT:38EDh)
46001                                       db      7,      "BUFFERS",      CONFIG_BUFFERS
46002                                       db      5,      "BREAK",        CONFIG_BREAK
46003                                       db      6,      "DEVICE",       CONFIG_DEVICE
46004                                       db      10,     "DEVICEHIGH",   CONFIG_DEVICEHIGH
46005                                       db      5,      "FILES",        CONFIG_FILES
46006                                       db      4,      "FCBS",         CONFIG_FCBS
46007                                       db      9,      "LASTDRIVE",    CONFIG_LASTDRIVE
46008                                       db      10,     "MULTITRACK",   CONFIG_MULTITRACK
46009                                       db      8,      "DRIVPARM",     CONFIG_DRIVPARM
46010                                       db      6,      "STACKS",       CONFIG_STACKS
46011                                       db      7,      "COUNTRY",      CONFIG_COUNTRY
46012                                       db      5,      "SHELL",        CONFIG_SHELL
46013                                       db      7,      "INSTALL",      CONFIG_INSTALL
46014                                       db      7,      "COMMENT",      CONFIG_COMMENT
46015                                       db      3,      "REM",          CONFIG_REM
46016                                       db      8,      "SWITCHES",     CONFIG_SWITCHES
46017                                       db      3,      "DOS",          CONFIG_DOS
46018                                       db      0
46019
46020                                   %endif
46021
46022                                   ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
46023                                   ; (SYSINIT:530Ch)
46024
46025                                   ; 13/04/2024 - Retro DOS v5.0
46026                                   ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:555Ah)
46027
46028                                   deviceparameters:
46029                                       ; A_DEVICEPARAMETERS <0,dev_3inch720kb,0,80>
46030                                   devp.specialfunc:  ; deviceparameters +
46031 00004DBE 00                           db      0       ; A_DEVICEPARAMETERS.DP_SPECIALFUNCTIONS
46032                                   devp.devtype:
46033 00004DBF 02                           db      2       ; A_DEVICEPARAMETERS.DP_DEVICETYPE
46034                                   devp.devattr:
46035 00004DC0 0000                         dw      0       ; A_DEVICEPARAMETERS.DP_DEVICEATTRIBUTES
46036                                   devp.cylinders:
46037 00004DC2 5000                         dw      80      ; A_DEVICEPARAMETERS.DP_CYLINDERS
46038
46039                                   ; 04/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
46040
46041                                       ;times  286     db 0
46042                                   devp.mediatype:             ; A_DEVICEPARAMETERS.DP_MEDIATYPE
46043 00004DC4 00                           db      0
46044                                   devp.bpb:           ; A_DEVICEPARAMETERS.DP_BPB
46045                                   devp.bps:           ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.BYTESPERSECTOR
46046 00004DC5 0000                         dw      0
46047                                   devp.secperclus:    ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERCLUSTER
46048 00004DC7 00                           db      0
46049 00004DC8 0000                         dw      0       ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.RESERVEDSECTORS
46050 00004DCA 00                           db      0       ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.NUMBEROFFATS
46051 00004DCB 0000                         dw      0       ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.ROOTENTRIES
46052                                   devp.totalsecs:             ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.TOTALSECTORS
46053 00004DCD 0000                         dw      0
46054                                   devp.mediaid:               ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.MEDIADESCRIPTOR
46055 00004DCF 00                           db      0
46056 00004DD0 0000                         dw      0       ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERFAT
46057                                   devp.spt:           ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERTRACK
46058 00004DD2 0000                         dw      0
46059                                   devp.heads:         ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.HEADS
46060 00004DD4 0000                         dw      0
46061
46062                                       ; 13/04/2024 - Retro DOS v5.0
46063                                       ; (PCDOS 7.1 IBMBIO.COM)
46064 00004DD6 00<rep 44h>                  times   68 db 0 ; PCDOS 7.1 (FAT32 BPB)
46065                                       ;;times 14 db 0 ; MSDOS 6.21
46066                                       ;dw     0       ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.HIDDENSECTORS
46067                                       ;dw     0
46068                                       ;dw     0       ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.BIGTOTALSECTORS
46069                                       ;dw     0
```

```
46070                                   ;times  6 db 0
46071
46072                                 devp.trktblents:
46073 00004E1A 0000                       dw      0       ; A_DEVICEPARAMETERS.DP_TRACKTABLEENTRIES
46074                                 devp.sectbl:            ; A_DEVICEPARAMETERS.DP_SECTORTABLE
46075 00004E1C 00<rep FCh>                times   252 db 0 ; MAX_SECTORS_IN_TRACK * A_SECTORTABLE.size
46076                                                     ; 63*4 bytes
46077
46078                                 ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
46079                                 ; (SYSINIT:5430h)
46080
46081                                 ; 13/04/2024 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
46082                                 ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:56B4h)
46083
46084 00004F18 0200                  hlim:       dw      2
46085 00004F1A 0900                  slim:       dw      9
46086
46087 00004F1C 00                    drive:      db      0
46088
46089                                 switches:
46090 00004F1D 0000                       dw      0
46091
46092                                 ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
46093                                 ; (SYSINIT:5437h)
46094
46095                                 ; the following are the recommended bpbs for the media that
46096                                 ; we know of so far.
46097
46098                                 ; 02/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
46099                                 ; MSDOS 5.0 IO.SYS - SYSINIT:3AA9h
46100
46101                                 ; 27/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
46102                                 ; PCDOS 7.1 IBMBIO.COM - SYSINIT:56BBh
46103
46104                                 ; 48 tpi diskettes
46105
46106 00004F1F 0002                  bpb48t:     dw      512
46107 00004F21 02                        db      2
46108 00004F22 0100                       dw      1
46109 00004F24 02                         db      2
46110 00004F25 7000                       dw      112
46111 00004F27 D002                       dw      2*9*40 ; 720
46112 00004F29 FD                         db      0FDh
46113 00004F2A 0200                       dw      2
46114 00004F2C 0900                       dw      9
46115 00004F2E 0200                       dw      2
46116 00004F30 00000000                   dd      0
46117 00004F34 00000000                       dd      0
46118                                     ; 27/12/2023
46119 00004F38 00<rep 1Ch>                times   28 db 0             ; FAT32 extensions (to BDS)
46120 00004F54 90                         db      90h
46121
46122                                 ; 96tpi diskettes
46123
46124 00004F55 0002                  bpb96t:     dw      512
46125 00004F57 01                         db      1
46126 00004F58 0100                       dw      1
46127 00004F5A 02                         db      2
46128 00004F5B E000                       dw      224
46129 00004F5D 6009                       dw      2*15*80 ; 2400
46130 00004F5F F9                         db      0F9h
46131 00004F60 0700                       dw      7
46132 00004F62 0F00                       dw      15
46133 00004F64 0200                       dw      2
46134 00004F66 00000000                   dd      0
46135 00004F6A 00000000                       dd      0
46136                                     ; 27/12/2023
46137 00004F6E 00<rep 1Ch>                times   28 db 0             ; FAT32 extensions (to BDS)
46138 00004F8A 90                         db      90h
46139
46140                                 ; 3 1/2 inch diskette bpb
46141
46142 00004F8B 0002                  bpb35:      dw      512
46143 00004F8D 02                         db      2
46144 00004F8E 0100                       dw      1
46145 00004F90 02                         db      2
46146 00004F91 7000                       dw      112
46147 00004F93 A005                       dw      2*9*80 ; 1440
46148 00004F95 F9                         db      0F9h
46149 00004F96 0300                       dw      3
46150 00004F98 0900                       dw      9
46151 00004F9A 0200                       dw      2
46152 00004F9C 00000000                   dd      0
46153 00004FA0 00000000                       dd      0
46154                                     ; 27/12/2023
46155 00004FA4 00<rep 1Ch>                times   28 db 0             ; FAT32 extensions (to BDS)
46156 00004FC0 90                         db      90h
46157
46158 00004FC1 0002                  bpb35h:     dw      512
46159 00004FC3 01                         db      1
46160 00004FC4 0100                       dw      1
46161 00004FC6 02                         db      2
46162 00004FC7 E000                       dw      224
46163 00004FC9 400B                       dw      2*18*80 ; 2880
46164 00004FCB F0                         db      0F0h
46165 00004FCC 0900                       dw      9
46166 00004FCE 1200                       dw      18
46167 00004FD0 0200                       dw      2
46168 00004FD2 00000000                   dd      0
46169 00004FD6 00000000                       dd      0
46170                                     ; 27/12/2023
46171 00004FDA 00<rep 1Ch>                times   28 db 0             ; FAT32 extensions (to BDS)
46172 00004FF6 90                         db      90h
46173
46174                                 ; m037 - BEGIN
46175
46176 00004FF7 0002                  bpb288:     dw      512
46177 00004FF9 02                         db      2
46178 00004FFA 0100                       dw      1
46179 00004FFC 02                         db      2
46180 00004FFD F000                       dw      240
46181 00004FFF 8016                       dw      2*36*80 ; 5760
46182 00005001 F0                         db      0F0h
46183 00005002 0900                       dw      9
46184 00005004 2400                       dw      36
46185 00005006 0200                       dw      2
46186 00005008 00000000                   dd      0
46187 0000500C 00000000                       dd      0
46188                                     ; 27/12/2023
46189 00005010 00<rep 1Ch>                times   28 db 0             ; FAT32 extensions (to BDS)
46190 0000502C 90                         db      90h
46191
46192                                 ; m037 - END
46193
```

```
                                    ; 12/05/2019
                                    
0000502D 90                         align 2
                                    
                                    ; 02/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; MSDOS 5.0 IO.SYS - SYSINIT:3B26h
                                    
                                    ; 13/04/2024 - Retro DOS v5.0
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:5738h)
                                    
0000502E [1F4F]                     bpbtable:   dw         bpb48t            ; 48tpi drives
00005030 [554F]                             dw     bpb96t              ; 96tpi drives
00005032 [8B4F]                             dw     bpb35               ; 3.5" drives
                                    ; the following are not supported, so default to 3.5" media layout
00005034 [8B4F]                             dw     bpb35               ; not used - 8" drives
00005036 [8B4F]                             dw     bpb35               ; not used - 8" drives
00005038 [8B4F]                             dw     bpb35               ; not used - hard files
0000503A [8B4F]                             dw     bpb35               ; not used - tape drives
0000503C [C14F]                             dw     bpb35h              ; 3-1/2" 1.44mb drive
0000503E [8B4F]                             dw     bpb35               ; ERIMO                         m037
00005040 [F74F]                             dw     bpb288              ; 2.88 MB diskette drives       m037
                                    
                                    switchlist:
00005042 08464853544449434E             db     8,"FHSTDICN"        ; preserve the positions of n and c.
                                    
                                    ;--------------------------------------------------------------------------
                                    ; Messages
                                    ;--------------------------------------------------------------------------
                                    
                                    ; 19/04/2019 - Retro DOS v4.0
                                    
                                    ; MSDOS 6.21 IO.SYS - SYSINIT:54D1h
                                    
0000504B 00                             db     0
                                    
                                    ; 02/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; MSDOS 5.0 IO.SYS - SYSINIT:3B44h
                                    
                                    ; 13/04/2024
                                    ; MSDOS 6.22 IO.SYS - SYSINIT:559Eh
                                    
                                    ; 13/04/2024 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
                                    ; PCDOS 7.1 IBMBIO.COM - SYSINIT:5756h
                                    
                                    badopm:
0000504C 0D0A                           db     0Dh,0Ah
0000504E 556E7265636F676E69-            db     'Unrecognized command in CONFIG.SYS'
00005057 7A656420636F6D6D61-
00005060 6E6420696E20434F4E-
00005069 4649472E535953
                                    crlfm:
00005070 0D0A24                         db     0Dh,0Ah,'$'
                                    badparm:
00005073 0D0A                           db     0Dh,0Ah
00005075 42616420636F6D6D61-            db     'Bad command or parameters - $'
0000507E 6E64206F7220706172-
00005087 616D6574657273202D-
00005090 2024
                                    badsiz_pre:
00005092 0D0A                           db     0Dh,0Ah
00005094 536563746F72207369-            db     'Sector size too large in file $'
0000509D 7A6520746F6F206C61-
000050A6 72676520696E206669-
000050AF 6C652024
                                    badld_pre:
000050B3 0D0A                           db     0Dh,0Ah
000050B5 426164206F72206D69-            db     'Bad or missing $'
000050BE 7373696E672024
                                    badcom:
000050C5 436F6D6D616E642049-            db     'Command Interpreter',0
000050CE 6E7465727072657465-
000050D7 7200
                                    badcountry:
000050D9 0D0A                           db     0Dh,0Ah
000050DB 496E76616C69642063-            db     'Invalid country code or code page',0Dh,0Ah,'$'
000050E4 6F756E74727920636F-
000050ED 6465206F7220636F64-
000050F6 6520706167650D0A24
                                    badcountrycom:
000050FF 0D0A                           db     0Dh,0Ah
00005101 4572726F7220696E20-            db     'Error in COUNTRY command',0Dh,0Ah,'$'
0000510A 434F554E5452592063-
00005113 6F6D6D616E640D0A24
                                    insufmemory:
0000511C 0D0A                           db     0Dh,0Ah
0000511E 496E73756666696369-            db     'Insufficient memory for COUNTRY.SYS file',0Dh,0Ah,'$'
00005127 656E74206D656D6F72-
00005130 7920666F7220434F55-
00005139 4E5452592E53595320-
00005142 66696C650D0A24
                                    badmem:
00005149 0D0A                           db     0Dh,0Ah
0000514B 436F6E666967757261-            db     'Configuration too large for memory',0Dh,0Ah,'$'
00005154 74696F6E20746F6F20-
0000515D 6C617267652066F72-
00005166 206D656D6F72790D0A-
0000516F 24
                                    badblock:
00005170 0D0A                           db     0Dh,0Ah
00005172 546F6F206D616E7920-            db     'Too many block devices',0Dh,0Ah,'$'
0000517B 626C6F636B20646576-
00005184 696365730D0A24
                                    badstack:
0000518B 0D0A                           db     0Dh,0Ah
0000518D 496E76616C69642053-            db     'Invalid STACK parameters',0Dh,0Ah,'$'
00005196 5441434B2070617261-
0000519F 6D65746572730D0A24
                                        ; 18/12/2022
                                    ;badorder:
                                        ;db    0Dh,0Ah
                                        ;db    'Incorrect order in CONFIG.SYS line $'
                                    errorcmd:
000051A8 4572726F7220696E20-            db     'Error in CONFIG.SYS line $'
000051B1 434F4E4649472E5359-
000051BA 53206C696E652024
                                    
                                    ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:566Eh)
                                    
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;%if 0
                                    
000051C2 4F4E                       OnOff:      db     'ON'
000051C4 4F4646                     OnOff2:     db     'OFF'
```

```
46287
46288                                   ; 04/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
46289                                   ; (SYSINIT:5673h)
46290                                 ;StartMsg:
46291                                 ;   db      'Starting MS-DOS...',0Dh,0Ah
46292                                 ;   db      0Ah,0
46293
46294                                   ; 17/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
46295                                   ; (SYSINIT:58F7h)
46296                                 StartMsg:
46297 000051C7 5374617274696E6720-        db      'Starting PC DOS...',0Dh,0Ah
46297 000051D0 504320444F532E2E2E-
46297 000051D9 0D0A
46298 000051DB 0A00                       db      0Ah,0
46299
46300                                 _$PauseMsg:
46301                                   ; 17/12/2023
46302                                   ;db     'Press any key to continue . . .',0Dh,0Ah,'$'
46303                                   ; 04/08/2023 (PCDOS 7.10 - IBMBIO.COM SYSINIT:590Dh)
46304 000051DD 507265737320616E79-        db      'Press any key to continue...',0Dh,0Ah,'$'
46304 000051E6 206B657920746F2063-
46304 000051EF 6F6E74696E75652E2E-
46304 000051F8 2E0D0A24
46305                                 _$CleanMsg:
46306                                   ;db     'MS-DOS is bypassing your CONFIG.SYS and AUTOEXEC.BAT files.',0Dh,0Ah,'$'
46307                                   ; 17/12/2023
46308 000051FC 504320444F53206973-        db      'PC DOS is bypassing your CONFIG.SYS and AUTOEXEC.BAT files.',0Dh,0Ah,'$'
46308 00005205 20627970617373696E-
46308 0000520E 6720796F757220434F-
46308 00005217 4E4649472E53595320-
46308 00005220 616E64204155544F45-
46308 00005229 5845432E4241542066-
46308 00005232 696C65732E0D0A24
46309                                 _$InterMsg:
46310                                   ;db     'MS-DOS will prompt you to confirm each CONFIG.SYS command.',0Dh,0Ah,'$'
46311                                   ; 17/12/2023
46312 0000523A 504320444F53207769-        db      'PC DOS will prompt you to confirm each CONFIG.SYS command.',0Dh,0Ah,'$'
46312 00005243 6C6C2070726F6D7074-
46312 0000524C 20796F7520746F2063-
46312 00005255 6F6E6669726D206561-
46312 0000525E 636820434F4E464947-
46312 00005267 2E53595320636F6D6D-
46312 00005270 616E642E0D0A24
46313                                 _$MenuHeader:
46314 00005277 0D0A                       db      0Dh,0Ah
46315                                     ; 17/12/2023
46316                                     ;db     '  MS-DOS 6.2 Startup Menu',0Dh,0Ah
46317                                     ;db     '  '
46318                                     ;times  23 db (0CDh)  ; ALT 205 ; '=======================' ; 06/08/2023
46319                                     ;db     0Dh,0Ah,'$'
46320                                     ; 04/08/2023 (PCDOS 7.10 - IBMBIO.COM SYSINIT:59A7h)
46321 00005279 2020504320444F5320-        db      '  PC DOS 7.1 Startup Menu',0Dh,0Ah
46321 00005282 372E312053746172742D-
46321 0000528B 7570204D656E750D0A
46322 00005294 2020                       db      '  '
46323 00005296 CD<rep 17h>                times   23 db (0CDh)  ; ALT 205 ; '=======================' ; 06/08/2023
46324 000052AD 0D0A24                     db      0Dh,0Ah,'$'
46325                                 _$MenuPrmpt:
46326 000052B0 2020456E7465722061-        db      '  Enter a choice: $'
46326 000052B9 2063686F6963653A20-
46326 000052C2 24
46327                                 _$StatusLine:
46328 000052C3 46353D427970617373-        db      'F5=Bypass startup files F8=Confirm each line of CONFIG.SYS '
46328 000052CC 207374617274757020-
46328 000052D5 66696C65732046383D-
46328 000052DE 436F6E6669726D2065-
46328 000052E7 616368206C696E6520-
46328 000052F0 6F6620434F4E464947-
46328 000052F9 2E53595320
46329 000052FE 616E64204155544F45-        db      'and AUTOEXEC.BAT [ ]$'
46329 00005307 5845432E424154205B-
46329 00005310 205D24
46330                                 _$InterPrmpt:
46331                                   ;db     ' [Y,N]?$'
46332                                   ; 13/04/2024
46333                                   ; 04/08/2023
46334 00005313 205B592C4E2C455343-        db      ' [Y,N,ESC]?$' ; PCDOS 7.1 - IBMBIO.COM
46334 0000531C 5D3F24
46335 0000531F 59455324               _$YES:      db      'YES$'
46336 00005323 4E4F2024               _$NO:       db      'NO $'
46337                                 _$TimeOut:
46338 00005327 54696D652072656D61-        db      'Time remaining: $'
46338 00005330 696E696E673A2024
46339                                 badcomprmpt:
46340                                   ;db     'Enter correct name of Command Interpreter (eg, C:\COMMAND.COM)'
46341                                   ; 13/04/2024 (PCDOS 7.1 IBMBIO.COM)
46342 00005338 456E74657220636F72-        db      'Enter correct name of Command Interpreter (for example, C:\COMMAND.COM)'
46342 00005341 72656374206E616D65-
46342 0000534A 206F6620436F6D6D61-
46342 00005353 6E6420496E74657270-
46342 0000535C 726574657220286F66-
46342 00005365 72206578616D706C65-
46342 0000536E 2C20433A5C434F4D4D-
46342 00005377 414E442E434F4D29
46343 0000537F 0D0A24                     db      0Dh,0Ah,'$'
46344                                 _$AutoPrmpt:
46345 00005382 50726F636573732041-        db      'Process AUTOEXEC.BAT [Y,N]?$'
46345 0000538B 55544F455845432E42-
46345 00005394 4154205B592C4E5D3F-
46345 0000539D 24
46346
46347                                 ;%endif ; 02/11/2022
46348
46349                                 ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
46350                                 ; (SYSINIT:5840h)
46351
46352                                 ; 02/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
46353                                 ; MSDOS 5.0 IO.SYS - SYSINIT:3CE0h
46354
46355                                 TooManyDrivesMsg:
46356 0000539E 5741524E494E472120-        db      'WARNING! Logical drives past Z: exist and will be ignored',0Dh,0Ah,'$'
46356 000053A7 4C6F676963616C2064-
46356 000053B0 726976657320706173-
46356 000053B9 74205A3A2065786973-
46356 000053C2 7420616E642077696C-
46356 000053CB 6C2062652069676E6F-
46356 000053D4 7265640D0A24
46357
46358                                 ; MSDOS 6.21 IO.SYS - SYSINIT:587Ch
46359                                   ;db     'Wrong DBLSPACE.BIN version',0Dh,0Ah,'$'
46360                                   ;db     7 dup(0)
46361
46362                                   ;times  7 db 0
46363                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
```

```
                                    ; MSDOS 5.0 IO.SYS - SYSINIT:3D1Ch
                                        ; 09/12/2022
                                        ;times 4 db 0

                                        ; 08/04/2024 - Retro DOS v5.0
                                    ; PCDOS 7.1 IBMBIO.COM - SYSINIT:5B0Bh
                                    baddblspace:
000053DA 526571756972656420-            db      'Required system component is not installed',0Dh,0Ah,'$',0
000053E3 73797374656D20636F-
000053EC 6D706F6E656E742069-
000053F5 73206E6F7420696E73-
000053FE 74616C6C65640D0A24-
00005407 00
                                        ;db     7 dup(0)

                                    ;--------------------------------------------------------------------------
                                                ; 09/12/2022
                                                ;db 0

                                    number3div  equ ($-SYSINIT$)
                                    number3mod  equ (number3div % 16)

                                    %if number3mod>0 & number3mod<16
00005408 00<rep 8h>                             times (16-number3mod) db 0
                                    %endif

                                    ;--------------------------------------------------------------------------
                                    ; 09/12/2022 - MSDOS 5.0 IO.SYS:3D20h ;;; SI_end = 3D20h for MSDOS 5.0 IO.SYS
                                    ;--------------------------------------------------------------------------

                                    ;MSDOS 6.21 IO.SYS - SYSINIT:5899h

                                    ;--------------------------------------------------------------------------
                                    ; 20/04/2019 - Retro DOS v4.0

                                    ; 09/12/2022
                                    ;
                                    ;bss_start:
                                    ;
                                    ;ABSOLUTE bss_start
                                    ;
                                    ;alignb 16

                                    SI_end:  ; SI_end equ $

                                    ;--------------------------------------------------------------------------

                                    ;sysinitseg ends

                                    ; ************************************************************************

                                    ; 04/01/2023 - MSDOS 6.21 SYSINIT:SI_end = SYSINIT:58A0h (IOSYS:9F46h)
                                    ; 09/12/2022 - MSDOS 5.0 SYSINIT:SI_end = SYSINIT:3D20h

                                    SYSINITSIZE equ SI_end - SYSINIT$
                                    DOSLOADSEG  equ SYSINITSEG+((SYSINITSIZE+15)/16)

                                    ;--------------------------------------------------------------------------
                                    ; End of Retro DOS 5.0 (PCDOS 7.1) IBMBIO.COM src by Erdogan Tan -21/04/2024-
                                    ;--------------------------------------------------------------------------
```