```
                        ; ****************************************************************************
                        ; IBMBIO7.S (PCDOS 7.1 IBMBIO.COM) - RETRO DOS 5.0 by ERDOGAN TAN - 12/09/2023
                        ; ----------------------------------------------------------------------------
                        ; Last Update: 19/01/2026 - 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 E9951B                 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:
```

```
4726
4727                                    ;  new code - let logical or clear carry and then set carry if ah!=0
4728                                    ;         and save a couple bytes while were at it.
4729
4730 00001747 58                                   pop     ax
4731                                               ;mov    ah, ds:74h      ; [disk_status1]
4732 00001748 8A267400                             mov     ah, [disk_status1]
4733 0000174C 08E4                                 or      ah, ah
4734 0000174E 7401                                 jz      short atd5
4735 00001750 F9                                   stc
4736                                    atd5:
4737 00001751 07                                   pop     es
4738 00001752 1F                                   pop     ds
4739 00001753 5F                                   pop     di
4740 00001754 5A                                   pop     dx
4741 00001755 59                                   pop     cx
4742 00001756 5B                                   pop     bx
4743 00001757 CA0200                               retf    2               ; far return, dropping flags
4744
4745                                    ; =============== S U B   R O U T I N E =======================================
4746
4747                                    ;***setcmd - set up cmd_block for the disk operation
4748                                    ;
4749                                    ;   entry:  (ds) = bios data segment.
4750                                    ;           (es:bx) in seg:000x form.
4751                                    ;           other registers as in int 13h call
4752                                    ;
4753                                    ;   exit:   cmd_block set up for disk read call.
4754                                    ;           control_byte set up for disk operation.
4755                                    ;           (al) = control byte modifier
4756                                    ;
4757                                    ;   sets the fields of cmd_block using the register contents
4758                                    ;   and the contents of the disk parameter block for the given drive.
4759                                    ;
4760                                    ;   warning: (ax) destroyed.
4761                                    ;           does direct calls to the at rom.
4762
4763                                    setcmd:             ; proc near
4764                                               ;mov    ds:43h,al       ; [cmd_block+sec_cnt]
4765                                               ; 16/10/2022
4766 0000175A A24300                             mov     [cmd_block+sec_cnt], al
4767                                               ;mov    byte ptr ds:48h, 20h ; [cmd_block+cmd_reg]
4768 0000175D C606480020                          mov     byte [cmd_block+cmd_reg], 20h ; assume function 02h (read)
4769 00001762 80FC02                              cmp     ah, 2
4770 00001765 7405                                jz      short setc1     ; cmd_reg = 20h       if function 02h       (read)
4771 00001767 C606480022                          mov     byte [cmd_block+cmd_reg], 22h
4772                                               ;mov    byte ptr ds:48h, 22h ; [cmd_block+cmd_reg]
4773                                                                       ; cmd_reg = 22h       if function 0Ah       (read long)
4774                                    setc1:
4775 0000176C 88C8                                mov     al, cl
4776 0000176E 243F                                and     al, 3Fh         ; mask sector number
4777                                               ;mov    ds:44h,al       ; [cmd_block+sec_num]
4778                                               ;mov    ds:45h,ch       ; [cmd_block+cyl_low]
4779 00001770 A24400                             mov     [cmd_block+sec_num], al ; mov [44h],al
4780 00001773 882E4500                           mov     [cmd_block+cyl_low], ch ; mov [45h],ch
4781 00001777 88C8                                mov     al, cl
4782 00001779 C0E806                              shr     al, 6           ; get two high bits of cylinder      number
4783                                               ;mov    ds:46h,al       ; [cmd_block+cyl_high]
4784 0000177C A24600                             mov     [cmd_block+cyl_high], al ; mov [46h],al
4785 0000177F 89D0                                mov     ax, dx
4786 00001781 C0E004                              shl     al, 4           ; drive number
4787 00001784 80E40F                              and     ah, 0Fh
4788 00001787 08E0                                or      al, ah          ; head number
4789 00001789 0CA0                                or      al, 0A0h        ; set ecc and 512 bytes       per sector
4790                                               ;mov    ds:47h,al       ; [cmd_block+drv_head]
4791 0000178B A24700                             mov     [cmd_block+drv_head], al  ; mov [47h],al
4792 0000178E 06                                  push    es
4793 0000178F 53                                  push    bx
4794 00001790 0E                                  push    cs
4795 00001791 E85C00                              call    get_vec
4796 00001794 268B4705                            mov     ax, [es:bx+5]  ; [es:bx+fdp_precomp]
4797                                                                       ; write pre-comp from disk parameters
4798 00001798 C1E802                              shr     ax, 2
4799                                               ;mov    ds:42h,al       ; [cmd_block+pre_comp]
4800 0000179B A24200                             mov     [cmd_block+pre_comp], al ; mov [42h],al
4801                                                                       ; only use low part
4802 0000179E 268A4708                            mov     al, [es:bx+8]  ; [es:bx+fdp_control]
4803                                                                       ; control byte modifier
4804 000017A2 5B                                  pop     bx
4805 000017A3 07                                  pop     es
4806                                               ;mov    ah, ds:76h      ; [control_byte]
4807 000017A4 8A267600                            mov     ah, [control_byte] ; mov ah,[76h]
4808 000017A8 80E4C0                              and     ah, 0C0h        ; keep disable retry bits
4809 000017AB 08C4                                or      ah, al
4810                                               ;mov    ds:76h,ah
4811 000017AD 88267600                            mov     [control_byte], ah ; mov [76h],al
4812 000017B1 C3                                  retn
4813
4814                                    ; =============== S U B   R O U T I N E =======================================
4815
4816                                    ;***docmd - carry out read operation to at hard disk
4817                                    ;
4818                                    ;   entry:  (es:bx) = address for read in data.
4819                                    ;           cmd_block set up for disk read.
4820                                    ;
4821                                    ;   exit:   buffer at (es:bx) contains data read.
4822                                    ;           disk_status1 set to error code (0 if success).
4823                                    ;
4824                                    ;
4825                                    ;   warning: (ax), (bl), (cx), (dx), (di) destroyed.
4826                                    ;           no check is made for dma boundary overrun.
4827                                    ;
4828                                    ;   effects: programs disk controller.
4829                                    ;           performs disk input.
4830
4831                                    docmd:              ; proc near
4832 000017B2 89DF                                mov     di, bx
4833 000017B4 0E                                  push    cs
4834 000017B5 E84000                              call    command
4835 000017B8 7535                                jnz     short doc3
4836                                    doc1:
4837 000017BA 0E                                  push    cs
4838 000017BB E84200                              call    waitt           ; wait for controller to complete read
4839 000017BE 752F                                jnz     short doc3
4840 000017C0 B90001                              mov     cx, 256         ; 256 words per sector
4841 000017C3 BAF001                              mov     dx, 1F0h        ; hf_port
4842 000017C6 FC                                  cld                     ; string op goes up
4843 000017C7 FA                                  cli                     ; disable interrupts
4844                                                                       ; (bug was forgetting this)
4845
4846                                    ;   M062 -- some of these old machines have intermittent failures
4847                                    ;           when the read is done at full speed. Instead of using
4848                                    ;           a string rep instruction, we'll use a loop. There is
4849                                    ;           a slight performance hit, but it only affects these
```

```
                                ;           very old machines with an exact date code match, and
                                ;           it makes said machines more reliable
                                ;
                                ;M062       repz    insw            ;read in sector
                                
                                rsct_loop:
000017C8 6D                     insw
000017C9 E2FD                   loop    rsct_loop
000017CB FB                     sti
                                ; 16/10/2022
000017CC F606480002             test    byte [cmd_block+cmd_reg], 02h
                                ;test   byte ptr ds:48h, 2 ; [cmd_block+cmd_reg]
                                                        ; (ds = 40h)
000017D1 7410                   jz      short doc2      ; no ecc bytes to read.
000017D3 0E                     push    cs
000017D4 E83100                 call    wait_drq        ; wait for controller to complete read
000017D7 7216                   jb      short doc3
000017D9 B90400                 mov     cx, 4           ; 4 bytes of ecc
000017DC BAF001                 mov     dx, 1F0h        ; hf_port
000017DF FA                     cli
000017E0 F36C                   rep insb                ; read in ecc
000017E2 FB                     sti
                                doc2:
000017E3 0E                     push    cs
000017E4 E82900                 call    check_status
000017E7 7506                   jnz     short doc3      ; operation failed
                                ;dec    byte ptr ds:43h         ; [cmd_block+sec_cnt]
000017E9 FE0E4300               dec     byte [cmd_block+sec_cnt]
000017ED 75CB                   jnz     short doc1      ; loop while more sectors to read
                                doc3:
000017EF C3                     retn
                                
                                ; =============== S U B   R O U T I N E =======================================
                                
                                ;***define where the rom routines are actually located
                                ;       in the buggy old AT BIOS that we might need to
                                ;       install a special level of int13 handler for
                                ; 16/10/2022
                                
                                romsegment  equ 0F000h  ; segment
                                romcommand  equ 2E1Eh   ; offset in romsegment
                                romwait             equ 2E7Fh   ; offset in romsegment
                                romwait_drq         equ 2EE2h   ; offset in romsegment
                                romcheck_status equ 2EF8h   ; offset in romsegment
                                romcheck_dma        equ 2F69h   ; offset in romsegment
                                romget_vec  equ 2F8Eh   ; offset in romsegment
                                romfret             equ 0FF65h  ; far return in rom
                                
                                ;***get_vec - get pointer to hard disk parameters.
                                ;
                                ;   entry:  (dl) = low bit has hard disk number (0 or 1).
                                ;
                                ;   exit:   (es:bx) = address of disk parameters table.
                                ;
                                ;   uses:   ax for segment computation.
                                ;
                                ;   loads es:bx from interrupt table in low memory, vector 46h (disk 0)
                                ;   or 70h (disk 1).
                                ;
                                ;   warning: (ax) destroyed.
                                ;           this does a direct call to the at rom.
                                
                                get_vec:    ; proc near
                                            ;push   0FF65h          ; romfret ; far       return in rom
                                            ;jmp    far ptr 0F000h:2F8Eh
                                            ; 16/10/2022
000017F0 6865FF                             push    romfret         ; far return in rom
000017F3 EA8E2F00F0                         jmp     romsegment:romget_vec
                                
                                ; =============== S U B   R O U T I N E =======================================
                                
                                ;***command - send contents of cmd_block to disk controller.
                                ;
                                ;   entry:  control_byte
                                ;           cmd_block - set up with values for hard disk controller.
                                ;
                                ;   exit:   disk_status1 = error code.
                                ;           nz if error, zr for no error.
                                ;
                                ;
                                ;   warning: (ax), (cx), (dx) destroyed.
                                ;           does a direct call to the at rom.
                                ;
                                ;   effects: programs disk controller.
                                
                                command:    ; proc near
                                            ;push   0FF65h          ; romfret ; far       return in rom
                                            ;jmp    far ptr 0F000h:2E1Eh
                                            ; 16/10/2022
000017F8 6865FF                             push    romfret         ; far return in rom
000017FB EA1E2E00F0                         jmp     romsegment:romcommand
                                
                                ; =============== S U B   R O U T I N E =======================================
                                
                                ;***waitt - wait for disk interrupt
                                ;
                                ;   entry:  nothing.
                                ;
                                ;   exit:   disk_status1 = error code.
                                ;           nz if error, zr if no error.
                                ;
                                ;
                                ;   warning: (ax), (bl), (cx) destroyed.
                                ;           does a direct call to the at rom.
                                ;
                                ;   effects: calls int 15h, function 9000h.
                                
                                waitt:              ; proc near
                                            ;push   0FF65h          ; romfret ; far       return in rom
                                            ;jmp    far ptr 0F000h:2E7Fh
                                            ; 16/10/2022
00001800 6865FF                             push    romfret         ; far return in rom
00001803 EA7F2E00F0                         jmp     romsegment:romwait
                                
                                ; =============== S U B   R O U T I N E =======================================
                                
                                ;***wait_drq - wait for data request.
                                ;
                                ;   entry:  nothing.
                                ;
                                ;   exit:   disk_status1 = error code.
                                ;           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:
```

```
5222 0000183E 0E                                push    cs
5223 0000183F 1F                                pop     ds
5224                                            ;cmp    byte [cs:inHMA], 0
5225 00001840 803E[0D00]00                      cmp     byte [inHMA], 0
5226 00001845 7405                              jz      short int6c
5227 00001847 BB[2A07]                          mov     bx, EnsureA20On
5228 0000184A FFD3                              call    bx
5229                                    int6c:
5230                                            ;push   cs
5231                                            ;pop    ds
5232 0000184C 8F06[F805]                        pop     word [int6c_ret_addr] ; pop off return address
5233 00001850 8F06[FA05]                        pop     word [int6c_ret_addr+2]
5234 00001854 9D                                popf
5235 00001855 E81300                            call    read_real_date ; get the date from the clock
5236 00001858 FA                                cli
5237 00001859 8936[8904]                        mov     [daycnt], si   ; update dos copy of date
5238 0000185D FB                                sti
5239 0000185E E8BB00                            call    read_real_time ; get the time from the      rtc
5240 00001861 FA                                cli
5241 00001862 B401                              mov     ah, 1
5242 00001864 CD1A                              int     1Ah             ; CLOCK - SET TIME OF DAY
5243                                                                     ; CX:DX = clock count
5244                                                                     ; Return: time of day set
5245 00001866 FB                                sti
5246                                            ;jmp    int6c_ret_addr ; long jump
5247                                            ; 16/10/2022
5248 00001867 FF2E[F805]                        jmp     far [int6c_ret_addr] ; long jump
5249
5250                                    ; =============== S U B   R O U T I N E =======================================
5251
5252                                    ;   read_real_date reads real-time clock for date and returns the number
5253                                    ;   of days elapsed since 1-1-80 in si
5254
5255                                    read_real_date:     ; proc near
5256 0000186B 50                                push    ax
5257 0000186C 51                                push    cx
5258 0000186D 52                                push    dx
5259 0000186E 30E4                              xor     ah, ah          ; throw away clock roll       over
5260 00001870 CD1A                              int     1Ah             ; CLOCK - GET TIME OF DAY
5261                                                                     ; Return: CX:DX       = clock count
5262                                                                     ; AL = 00h if clock was       read or written (via AH=0,1) since the previous
5263                                                                     ; midnight
5264                                                                     ; Otherwise, AL       > 0
5265 00001872 5A                                pop     dx
5266 00001873 59                                pop     cx
5267 00001874 58                                pop     ax
5268 00001875 50                                push    ax
5269 00001876 53                                push    bx
5270 00001877 51                                push    cx
5271 00001878 52                                push    dx
5272                                            ;mov    word [cs:daycnt2], 1
5273                                            ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
5274                                            ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:187Ah
5275 00001879 C706[0006]0100                    mov     word [daycnt2], 1
5276                                                                     ; REAL TIME CLOCK ERROR       FLAG (+1 DAY)
5277 0000187F B404                              mov     ah, 4
5278 00001881 CD1A                              int     1Ah             ; CLOCK - READ DATE FROM REAL TIME CLOCK (AT,XT286,CONV,PS)
5279                                                                     ; Return: DL = day in BCD
5280                                                                     ; DH = month in       BCD
5281                                                                     ; CL = year in BCD
5282                                                                     ; CH = century (19h or 20h)
5283 00001883 7303                              jnb     short read_ok
5284 00001885 E98500                            jmp     r_d_ret
5285                                    ;-----------------------------------------------------------------------------
5286
5287                                    read_ok:
5288 00001888 882E[FC05]                        mov     [bin_date_time], ch
5289 0000188C 880E[FD05]                        mov     [bin_date_time+1], cl
5290 00001890 8836[FE05]                        mov     [bin_date_time+2], dh
5291 00001894 8816[FF05]                        mov     [bin_date_time+3], dl
5292                                            ;mov    word [cs:daycnt2], 2 ; READ OF R-T CLOCK SUCCESSFUL
5293                                            ; 08/08/2023
5294                                            ;mov    byte [daycnt2], 2
5295 00001898 FE06[0006]                        inc     byte [daycnt2] ; 2
5296 0000189C E83601                            call    bcd_verify     ; verify bcd values in range
5297 0000189F 726C                              jb      short r_d_ret  ; some value out of range
5298                                            ;mov    word [cs:daycnt2], 3
5299                                            ; 08/08/2023
5300                                            ;mov    byte [daycnt2], 3
5301 000018A1 FE06[0006]                        inc     byte [daycnt2] ; 3
5302 000018A5 E8DD00                            call    date_verify
5303 000018A8 7263                              jb      short r_d_ret
5304                                            ;mov    word [cs:daycnt2], 0
5305                                            ; 08/08/2023
5306 000018AA C606[0006]00                      mov     byte [daycnt2], 0
5307 000018AF E8A300                            call    in_bin
5308 000018B2 A0[FD05]                          mov     al, [bin_date_time+1]
5309 000018B5 98                                cbw
5310 000018B6 803E[FC05]14                      cmp     byte [bin_date_time], 20 ; 20th century?
5311 000018BB 7503                              jnz     short century_19 ; no
5312 000018BD 83C064                            add     ax, 100         ; add in a century
5313                                    century_19:
5314 000018C0 83E850                            sub     ax, 80          ; subtract off 1-1-80
5315 000018C3 B104                              mov     cl, 4           ; leap year every 4
5316 000018C5 F6F1                              div     cl              ; al= # leap year blocks, ah= remainder
5317 000018C7 88E3                              mov     bl, ah          ; save odd years
5318 000018C9 98                                cbw                     ; zero ah
5319 000018CA B9B505                            mov     cx, 1461        ; 366+(3*365)
5320                                                                    ; # of days in leap year blocks
5321 000018CD F7E1                              mul     cx
5322                                            ;mov    [cs:daycnt2], ax ; SAVE COUNT OF DAYS
5323                                            ; 08/08/2023
5324 000018CF A3[0006]                          mov     [daycnt2], ax
5325 000018D2 88D8                              mov     al, bl          ; get odd years       count
5326 000018D4 98                                cbw
5327 000018D5 09C0                              or      ax, ax
5328 000018D7 740B                              jz      short leap_year
5329 000018D9 B96D01                            mov     cx, 365         ; days in year
5330 000018DC F7E1                              mul     cx
5331                                            ;add    [cs:daycnt2], ax ; ADD ON DAYS IN ODD YEARS
5332                                            ; 08/08/2023
5333 000018DE 0106[0006]                        add     [daycnt2], ax
5334 000018E2 EB07                              jmp     short leap_adjustment ;       account for leap year
5335                                                                    ; possibly account for a leap day
5336                                    ;-----------------------------------------------------------------------------
5337
5338                                    leap_year:
5339 000018E4 803E[FE05]02                      cmp     byte [bin_date_time+2], 2 ; is       month february?
5340 000018E9 7604                              jbe     short no_leap_adjustment ; jan or feb. no leap day yet.
5341                                    leap_adjustment:
5342                                            ;inc    word [cs:daycnt2] ; account for leap day
5343                                            ; 08/08/2023
5344 000018EB FF06[0006]                        inc     word [daycnt2]
5345                                    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
                                        ; 19/01/2026 (BugFix)
000018FF 740C                           jz      short r_d_ret

                                        ; 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
00001901 B400                           mov     ah, 0
00001903 BE[8F04]                       mov     si, month_tab
                                r_d_sum_loop:
00001906 AC                             lodsb
00001907 0106[0006]                     add     [daycnt2], ax
0000190B E2F9                           loop    r_d_sum_loop
                                r_d_ret:
                                        ;mov    si, [cs:daycnt2]
                                        ; 08/08/2023
0000190D 8B36[0006]                     mov     si, [daycnt2]
00001911 5A                             pop     dx
00001912 59                             pop     cx
00001913 5B                             pop     bx
00001914 58                             pop     ax
00001915 C3                             retn

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

                                r_t_retj:
00001916 31C9                           xor     cx, cx
00001918 31D2                           xor     dx, dx
0000191A 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
0000191C B402                           mov     ah, 2
0000191E 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
00001920 72F4                           jb      short r_t_retj
00001922 882E[FC05]                     mov     [bin_date_time], ch ; hours
00001926 880E[FD05]                     mov     [bin_date_time+1], cl ; minutes
0000192A 8836[FE05]                     mov     [bin_date_time+2], dh ; seconds
0000192E C606[FF05]00                   mov     byte [bin_date_time+3], 0 ; unused for time
00001933 E89F00                         call    bcd_verify
00001936 72DE                           jb      short r_t_retj
00001938 E88500                         call    time_verify
0000193B 72D9                           jb      short r_t_retj
0000193D E81500                         call    in_bin          ; from bcd to bin
00001940 8A2E[FC05]                     mov     ch, [bin_date_time]
00001944 8A0E[FD05]                     mov     cl, [bin_date_time+1]
00001948 8A36[FE05]                     mov     dh, [bin_date_time+2]
0000194C 8A16[FF05]                     mov     dl, [bin_date_time+3]
                                        ; 16/10/2022
                                        ; 17/09/2022
                                        ; 31/05/2019
00001950 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:
00001954 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
00001955 A0[FC05]                       mov     al, [bin_date_time] ; century or hours
00001958 E81F00                         call    bcd_to_bin
0000195B A2[FC05]                       mov     [bin_date_time], al
0000195E A0[FD05]                       mov     al, [bin_date_time+1] ; years or minutes
00001961 E81600                         call    bcd_to_bin
00001964 A2[FD05]                       mov     [bin_date_time+1], al
00001967 A0[FE05]                       mov     al, [bin_date_time+2] ; months or seconds
0000196A E80D00                         call    bcd_to_bin
0000196D A2[FE05]                       mov     [bin_date_time+2], al
00001970 A0[FF05]                       mov     al, [bin_date_time+3] ; days (not used for time)
00001973 E80400                         call    bcd_to_bin
00001976 A2[FF05]                       mov     [bin_date_time+3], al
00001979 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
0000197A 88C4                           mov     ah, al
0000197C 240F                           and     al, 0Fh
0000197E B104                           mov     cl, 4
00001980 D2EC                           shr     ah, cl
00001982 D50A                           aad
00001984 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
```

```
5470                                    date_verify:       ; proc near
5471 00001985 803E[FC05]20                              cmp     byte [bin_date_time], 20h ; century check
5472 0000198A 7732                                      ja      short date_error
5473 0000198C 740E                                      jz      short century_20 ; jmp in 21th century
5474 0000198E 803E[FC05]19                              cmp     byte [bin_date_time], 19h ; century check
5475                                                    ;jb     short date_error
5476                                                    ; 12/12/2022
5477 00001993 722A                                      jb      short date_err2
5478 00001995 803E[FD05]80                              cmp     byte [bin_date_time+1], 80h ; year check
5479                                                    ;jb     short date_error
5480                                                    ; 12/12/2022
5481 0000199A 7223                                      jb      short date_err2
5482                                    century_20:
5483 0000199C 803E[FD05]99                              cmp     byte [bin_date_time+1], 99h ; year check
5484 000019A1 771B                                      ja      short date_error
5485 000019A3 803E[FE05]12                              cmp     byte [bin_date_time+2], 12h ; month check
5486 000019A8 7714                                      ja      short date_error
5487 000019AA 803E[FE05]00                              cmp     byte [bin_date_time+2], 0
5488                                                    ;jbe    short date_error
5489 000019AF 760D                                      jna     short date_error
5490 000019B1 803E[FF05]31                              cmp     byte [bin_date_time+3], 31h ; day check
5491 000019B6 7706                                      ja      short date_error
5492                                                    ;cmp    byte [bin_date_time+3], 0 ; day check
5493                                                    ;;jbe   short date_error
5494                                                    ;jna    short date_error
5495                                                    ; 12/12/2022
5496                                                    ; cf=0
5497                                                    ;clc
5498                                                    ; 12/12/2022
5499 000019B8 803E[FF05]01                              cmp     byte [bin_date_time+3], 1 ; day check
5500 000019BD C3                                        retn
5501                                    ;---------------------------------------------------------------------------
5502
5503                                    date_error:
5504 000019BE F9                                        stc
5505                                    date_err2:
5506 000019BF C3                                        retn
5507
5508                                    ; =============== S U B   R O U T I N E =======================================
5509
5510                                    ; time_verify very loosely checks bcd date values to be in range
5511                                    ; in bin_date_time
5512
5513                                    time_verify:       ; proc near
5514 000019C0 803E[FC05]24                              cmp     byte [bin_date_time], 24h ; hour check
5515 000019C5 770C                                      ja      short time_error
5516 000019C7 803E[FD05]59                              cmp     byte [bin_date_time+1], 59h ; minute check
5517 000019CC 7705                                      ja      short time_error
5518                                                    ; 12/12/2022h
5519                                                    ;cmp    byte [bin_date_time+2], 59h ; second check
5520                                                    ;ja     short time_error
5521                                                    ;clc
5522                                                    ;retn
5523                                                    ; 12/12/2022
5524 000019CE 803E[FE05]5A                              cmp     byte  [bin_date_time+2], 5Ah
5525                                    time_error:
5526                                    bv_error:
5527 000019D3 F5                                        cmc     ; cf=0 -> cf=1, cf=1 -> cf=0
5528 000019D4 C3                                        retn
5529
5530                                    ; ---------------------------------------------------------------------------
5531
5532                                    ;time_error:
5533                                                    ;stc
5534                                                    ;retn
5535
5536                                    ; =============== S U B   R O U T I N E =======================================
5537
5538                                    ;   bcd_verify checks values in bin_date_time to be valid
5539                                    ;   bcd numerals.  carry set if any nibble out of range
5540
5541                                    bcd_verify: ; proc near
5542 000019D5 B90400                                    mov     cx, 4           ; 4 bytes to check
5543 000019D8 BB[FC05]                                  mov     bx, bin_date_time
5544                                    bv_loop:
5545 000019DB 8A07                                      mov     al, [bx]        ; get a bcd number (0..99)
5546 000019DD 88C4                                      mov     ah, al
5547 000019DF 250FF0                                    and     ax, 0F00Fh      ; 10's place in high ah, 1's in al
5548                                                                            ; is 1's place in range?
5549 000019E2 3C0A                                      cmp     al, 10
5550 000019E4 77ED                                      ja      short bv_error ; jmp out of range
5551 000019E6 D0EC                                      shr     ah, 1
5552 000019E8 D0EC                                      shr     ah, 1
5553 000019EA D0EC                                      shr     ah, 1
5554 000019EC D0EC                                      shr     ah, 1
5555 000019EE 80E40F                                    and     ah, 0Fh         ; get rid of any erroneous bits
5556 000019F1 80FC0A                                    cmp     ah, 10          ; is 10's place in range
5557 000019F4 77DD                                      ja      short bv_error ; jmp out of range
5558 000019F6 43                                        inc     bx              ; next byte
5559 000019F7 49                                        dec     cx
5560 000019F8 75E1                                      jnz     short bv_loop
5561 000019FA F8                                        clc                     ; set success flag
5562 000019FB C3                                        retn
5563                                    ; ---------------------------------------------------------------------------
5564
5565                                                    ; 12/12/2022
5566                                    ;bv_error:
5567                                                    ;stc                    ; set error flag
5568                                                    ;retn
5569
5570                                    ; ---------------------------------------------------------------------------
5571
5572                                    endk09:
5573
5574                                    ; ---------------------------------------------------------------------------
5575
5576                                    ;---------------------------------------------------------------------------
5577                                    ;                                                                         :
5578                                    ;   System initialization                                                :
5579                                    ;                                                                         :
5580                                    ;   The entry conditions are established by the bootstrap                      :
5581                                    ;   loader and are considered unknown. The following jobs                     :
5582                                    ;   will be performed by this module:                                     :
5583                                    ;                                                                         :
5584                                    ;   1.      All device initialization is performed                            :
5585                                    ;   2.      A local stack is set up and DS:SI are set                      :
5586                                    ;           to point to an initialization table. Then                      :
5587                                    ;           an inter-segment call is made to the first                     :
5588                                    ;           byte of the dos                                                :
5589                                    ;   3.      Once the dos returns from this call the ds                     :
5590                                    ;           register has been set up to point to the start                    :
5591                                    ;           of free memory. The initialization will then                   :
5592                                    ;           load the command program into this area                        :
5593                                    ;           beginning at 100 hex and transfer control to                   :
```

```
5594                                    ;           this program.                                           :
5595                                    ;                                                                   :
5596                                    ;-------------------------------------------------------------------
5597
5598                                    ; 01/10/2022
5599                                    ; 08/01/2018 - Retro DOS v4.0
5600
5601                                    ; drvfat must be the first location of freeable space!
5602
5603                                    align 2
5604                                                ;db 90h
5605
5606                                    ; 09/12/2023 - Retro DOS v5.0 IO.SYS (IBMBIO.COM 7.1)
5607                                    ; (PCDOS 7.1 IBMBIO.COM, BIOSDATA:1A0Ch)
5608
5609                                    ; 30/12/2022
5610                                    ; (MSDOS 6.21 IO.SYS, BIOSDATA:16D6h)
5611
5612 000019FC 0000                      drvfat:             dw 0                    ; drive and fat id of dos
5613                                    ; 09/12/2023
5614                                    ;bios_l:    dw 0                    ; first sector of data (low word)
5615                                    ;bios_h:    dw 0                    ; first sector of data (high word)
5616                                    First_Data_Sector:
5617 000019FE 0000                                  dw 0
5618 00001A00 0000                                  dw 0
5619 00001A02 0000                      doscnt:             dw 0                    ; how many sectors to read
5620                                    ;fbigfat:   db 0                    ; flags for drive
5621 00001A04 0000                      fatloc:             dw 0                    ; seg addr of fat sector
5622 00001A06 0000                      init_bootseg:       dw 0                    ; seg addr of buffer for reading boot record
5623                                    ; 09/12/2023
5624 00001A08 00                        fbigfat:    db 0                    ; flags for drive
5625 00001A09 80                        rom_drv_num:        db 80h                  ; rom drive number
5626 00001A0A 0002                      md_sectorsize:      dw 200h                 ; used by get_fat_sector proc.
5627                                    ; 12/12/2023
5628                                    ;temp_cluster:      dw 0                    ; used by get_fat_sector proc.
5629 00001A0C FFFF                      last_fat_sec_num:   dw 0FFFFh               ; used by get_fat_sector proc.
5630
5631                                    ; the following two bytes are used to save the info returned by int 13, ah = 8
5632                                    ; call to determine drive parameters.
5633
5634 00001A0E 02                        num_heads:  db 2    ; dw 2          ; number of heads returned by rom
5635 00001A0F 00                                    db 0    ; 09/12/2023
5636                                    ;sec_trk:   db 9                    ; sec/trk returned by rom
5637 00001A10 28                        num_cyln:   db 40   ; dw 40         ; number of cylinders returned by rom
5638 00001A11 00                                    db 0    ; 09/12/2023
5639                                    ; 09/12/2023
5640 00001A12 09                        sec_trk:    db 9                    ; sec/trk returned by rom
5641 00001A13 00                        fakefloppydrv:      db 0                    ; if 1, then no diskette drives       in the system.
5642
5643                                    ; 09/12/2023 - Retro DOS v5.0 IO.SYS/IBMBIO.COM
5644                                    Orig_Int1Eh_Table:
5645 00001A14 0000                                  dw 0
5646 00001A16 0000                                  dw 0
5647
5648                                    ; ---------------------------------------------------------------------------
5649
5650                                    ; 09/12/2023
5651                                    %if 0
5652
5653                                    disktable:  dw 512, 0100h, 64, 0    ; warning !!! old values
5654                                                dw 2048, 0201h, 112, 0
5655                                                dw 8192, 0402h, 256, 0
5656                                                dw 32680, 0803h, 512, 0         ; warning !!! old values
5657                                                dw 65535, 1004h, 1024, 0
5658                                                                        ; default disktable under
5659                                                                        ; the assumption of total fat size <= 128 kb,
5660                                                                        ; and the maximum size of fat entry = 16 bit.
5661                                    %endif
5662
5663                                    ; 09/12/2023 - Retro DOS v5.0 IO.SYS (IBMBIO.COM 7.1)
5664                                    ; (PCDOS 7.1 IBMBIO.COM, BIOSDATA:1A2Ah)
5665
5666                                                ; 09/12/2023
5667                                                ; 08/08/2023
5668                                                ; disktable.totalsectors: resw 1 ; high word
5669                                                ;                         resw 1 ; low word
5670                                                ; disktable.shiftcount:   resb 1
5671                                                ; disktable.secperclus:   resb 1
5672                                                ; disktable.rdirentries:  resw 1
5673                                                ; disktable.bigflag:      resw 1
5674 00001A18 0000A87F0308000200-       disktable2: dw 0, 32680, 0803h, 512, 0 ; for compatibility.
5674 00001A21 00
5675                                                                        ; (32680 sectors, 16340 KB)
5676 00001A22 040000000204000240-                   dw 4, 0, 0402h, 512, 40h    ; covers upto 134 mb media.
5676 00001A2B 00
5677                                                                        ; fbig = 40h  ; (40000h sectors = 128 MB)
5678 00001A2C 080000000308000240-                   dw 8, 0, 0803h, 512, 40h    ; upto 268 mb ; (80000h sectors = 256 MB)
5678 00001A35 00
5679 00001A36 100000000410000240-                   dw 16, 0, 1004h, 512, 40h  ; upto 536 mb ; (100000h sectors = 512 MB)
5679 00001A3F 00
5680 00001A40 200000000520000240-                   dw 32, 0, 2005h, 512, 40h  ; upto 1072 mb ; (200000h sectors = 1024 MB)
5680 00001A49 00
5681 00001A4A 400000000640000240-                   dw 64, 0, 4006h, 512, 40h  ; upto 2144 mb ; (400000h sectors = 2048 MB)
5681 00001A53 00
5682                                                ; 09/12/2023
5683                                                ;dw 128, 0, 8007h, 512, 40h ; upto 4288 mb ; (800000h sectors = 4096 MB)
5684 00001A54 FFFFFFFF0308000060-                   dw 0FFFFh, 0FFFFh, 0803h, 0, 60h ; FAT32 (> 2144MB)
5684 00001A5D 00
5685                                                                        ; (fbig and fbigbig flags are set)
5686
5687                                    ; ---------------------------------------------------------------------------
5688
5689                                    ;*******************************************************
5690                                    ;variables for mini disk initialization
5691                                    ;*******************************************************
5692
5693                                    ; 01/10/2022
5694                                    ; [ Note: Minidisk == logical dos drive (in extended dos partition) ]
5695
5696 00001A5E 00                        rom_minidisk_num: db 0                  ; temp variable       for phys unit
5697 00001A5F 00                        hnum:               db 0                ; real number of hardfiles
5698 00001A60 [3C05]                    last_dskdrv_table:  dw dskdrvs          ; index into dskdrv table
5699 00001A62 [4C08]                    end_of_bdss:        dw bdss             ; offset value of the ending address
5700                                                                        ; of bds table. needed to figure out
5701                                                                        ; the dosdatasg address.
5702 00001A64 0000                      mini_hdlim: dw 0
5703 00001A66 0000                      mini_seclim:        dw 0
5704
5705                                    ; 19/12/2023
5706                                    ; 09/12/2023
5707                                    ; (PCDOS 7.1 IBMBIO.COM, BIOSDATA:1A7Ah)
5708                                    ;ld_p_number:       dw 2BADh            ; (for 'find_mini_partition' proc)
5709
5710                                    ;end of mini disk init variables **********************
```

```
5711
5712                                    ; ----------------------------------------------------------------------------
5713
5714 00001A68 30312F31302F383400        bios_date:  db '01/10/84',0       ; used for checking at rom bios      date.
5715
5716                                    ; 13/12/2022
5717                                    %if 0
5718
5719                                    ;align 2
5720                                                db  90h
5721
5722                                    ; the following are the recommended bpbs for the media that we know of so far.
5723
5724                                    ;struc bpbx
5725                                    ;   resw 1 ; 512
5726                                    ;   resb 1
5727                                    ;   resw 1 ; 1
5728                                    ;   resb 1 ; 2
5729                                    ;   resw 1
5730                                    ;   resw 1
5731                                    ;   resb 1
5732                                    ;   resw 1
5733                                    ;   resw 1
5734                                    ;   resw 1 ; 2
5735                                    ;   resw 1
5736                                    ;   resw 1 ; hidden sector high
5737                                    ;   resd 1 ; extended total sectors
5738                                    ;.size:
5739                                    ;endstruc
5740
5741                                    ; 08/01/2019 - Retro DOS v4.0
5742
5743                                    ; 20/04/2019
5744
5745                                    ; 01/10/2022 - Retro DOS v4.0 (MSDOS 5.0) IO.SYS
5746
5747                                    ; 09/12/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
5748                                    ; (PCDOS 7.1 IBMBIO.COM, BIOSDATA:1A86h)
5749
5750                                    ; 09/12/2022
5751                                    BPB48T:
5752                                    ;bpb48t:    ; bpbx <512, 2, 1, 2, 112, 720, 0FDh, 2, 9, 2, 0, 0, 0, 0>
5753                                                ; 48 tpi diskettes    ;
5754                                                dw      512             ; physical sector size in bytes
5755                                                db      2               ; sectors/allocation unit
5756                                                dw      1               ; reserved sectors for dos
5757                                                db      2               ; number of allocation tables
5758                                                dw      112             ; number of directory entries
5759                                                dw      720 ; 2*9*40    ; number of sectors (at 512 bytes each)
5760                                                db      0FDh            ; media descriptor
5761                                                dw      2               ; number of fat sectors
5762                                                dw      9               ; sectors per track
5763                                                dw      2               ; heads
5764                                                dw      0               ; hidden sector count (low word)
5765                                                dw      0               ; hidden sector (high)
5766                                                dw      0               ; number of sectors (low)
5767                                                dw      0               ; number of sectors (high)
5768                                                ; 09/12/2023
5769                                                ; FAT32 extensions (to BDS)
5770                                                times  28 db 0
5771                                                ;
5772                                                db 90h
5773                                    ;align 2
5774                                    BPB96T:
5775                                    ;bpb96t:    ; bpbx <512, 1, 1, 2, 224, 2400, 0F9h, 7, 15, 2, 0, 0, 0, 0>
5776                                                ; 96 tpi diskettes    ;
5777                                                dw      512             ; physical sector size in bytes
5778                                                db      1               ; sectors/allocation unit
5779                                                dw      1               ; reserved sectors for dos
5780                                                db      2               ; number of allocation tables
5781                                                dw      224             ; number of directory entries
5782                                                dw      2400 ; 2*15*80  ; number of sectors (at 512 bytes each)
5783                                                db      0F9h            ; media descriptor
5784                                                dw      7               ; number of fat sectors
5785                                                dw      15              ; sectors per track
5786                                                dw      2               ; heads
5787                                                dw      0               ; hidden sector count (low word)
5788                                                dw      0               ; hidden sector (high)
5789                                                dw      0               ; number of sectors (low)
5790                                                dw      0               ; number of sectors (high)
5791                                                ; 09/12/2023
5792                                                ; FAT32 extensions (to BDS)
5793                                                times  28 db 0
5794                                                ;
5795                                                db 90h
5796                                    ;align 2
5797                                    BPB35:
5798                                    ;bpb35:             ; bpbx <512, 2, 1, 2, 112, 1440, 0F9h, 3, 9, 2, 0, 0, 0, 0>
5799                                                ; 3.5" diskettes - 720 KB ;
5800                                                dw      512             ; physical sector size in bytes
5801                                                db      2               ; sectors/allocation unit
5802                                                dw      1               ; reserved sectors for dos
5803                                                db      2               ; number of allocation tables
5804                                                dw      112             ; number of directory entries
5805                                                dw      1440 ; 2*9*80   ; number of sectors (at 512 bytes each)
5806                                                db      0F9h            ; media descriptor
5807                                                dw      3               ; number of fat sectors
5808                                                dw      9               ; sectors per track
5809                                                dw      2               ; heads
5810                                                dw      0               ; hidden sector count (low word)
5811                                                dw      0               ; hidden sector (high)
5812                                                dw      0               ; number of sectors (low)
5813                                                dw      0               ; number of sectors (high)
5814                                                ; 09/12/2023
5815                                                ; FAT32 extensions (to BDS)
5816                                                times  28 db 0
5817                                                ;
5818                                                db 90h
5819                                    ;align 2
5820
5821                                    ;align 2
5822                                    ;BPB144:
5823                                    ;bpb144:    ; Retro DOS v4.0 feature only !      ; 1.44MB diskettes
5824                                    ;
5825                                    ;           dw      512             ; physical sector size in bytes
5826                                    ;           db      1               ; sectors/allocation unit
5827                                    ;           dw      1               ; reserved sectors for dos
5828                                    ;           db      2               ; number of allocation tables
5829                                    ;           dw      224             ; number of directory entries
5830                                    ;           dw      2880 ; 2*18*80  ; number of sectors (at 512 bytes each)
5831                                    ;           db      0F0h            ; media descriptor
5832                                    ;           dw      9               ; number of fat sectors
5833                                    ;           dw      18              ; sectors per track
5834                                    ;           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
0000:1A71 [CA00]                                dw BC_RETF              ; dw 0CAh

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

                                call_bios_code:    ; proc far
0000:1A73 2EFF36[711A]                          push    word [cs:addr_of_bcretf]
                                                                        ; set up near return to far return
0000:1A78 2EFF36[0406]                          push    word [cs:cdev+2] ; push Bios_Code segment
0000:1A7D 55                                    push    bp              ; save offset of utility function
0000:1A7E CB                                    retf                    ; far jump to (DOS)BIOS code

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

                                            ; 09/12/2023 - Retro DOS v5.0 IBMBIO.COM/IO.SYS
                                            ; 20/12/2022
0000:1A7F 00                    flp_drvs:   db 0
                                            ; 11/12/2023
                                            ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:1B81h)
                                firstcluster_hw:
0000:1A80 0000                              dw 0    ; 06/04/2024
0000:1A82 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)
```

```
5959
5960                                    cd_boot_option:
5961 00001A83 50                                push    ax
5962 00001A84 1E                                push    ds
5963 00001A85 06                                push    es
5964 00001A86 52                                push    dx
5965                                    cdbo_1:
5966 00001A87 B401                              mov     ah, 1
5967 00001A89 CD16                              int     16h                     ; KEYBOARD - status
5968 00001A8B 7406                              jz      short cdbo_2
5969 00001A8D 30E4                              xor     ah, ah
5970 00001A8F CD16                              int     16h                     ; KEYBOARD - READ CHAR FROM BUFFER, WAIT IF EMPTY
5971                                                                            ; Return: AH = scan code, AL = character
5972 00001A91 EBF4                              jmp     short cdbo_1
5973                                    cdbo_2:
5974 00001A93 0E                                push    cs
5975 00001A94 1F                                pop     ds
5976 00001A95 BE[6D1B]                          mov     si, cd_boot_msg                 ; "Press the ENTER key to boot from CD"...
5977 00001A98 AC                                lodsb
5978                                    cdbo_3:
5979 00001A99 BB0700                            mov     bx, 7
5980 00001A9C B40E                              mov     ah, 0Eh
5981 00001A9E CD10                              int     10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
5982                                                                            ; AL = character, BH = display page (alpha modes)
5983                                                                            ; BL = foreground color (graphics modes)
5984 00001AA0 AC                                lodsb
5985 00001AA1 08C0                              or      al, al
5986 00001AA3 75F4                              jnz     short cdbo_3
5987 00001AA5 B84000                            mov     ax, 40h
5988 00001AA8 8ED8                              mov     ds, ax
5989                                            ;mov    bx, [6Ch]               ; 0:46Ch = Daily timer counter (4 bytes)
5990                                            ; 09/12/2023
5991 00001AAA 8B166C00                          mov     dx, [6Ch]
5992 00001AAE 8B366E00                          mov     si, [6Eh]
5993                                    wait_for_key:
5994                                            ;push   bx
5995                                            ;mov    bx, 7
5996                                            ; bx = 7
5997 00001AB2 B8080E                            mov     ax, 0E08h
5998 00001AB5 CD10                              int     10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
5999                                                                            ; AL = character, BH = display page (alpha modes)
6000                                                                            ; BL = foreground color (graphics modes)
6001 00001AB7 B8200E                            mov     ax, 0E20h
6002 00001ABA CD10                              int     10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
6003                                                                            ; AL = character, BH = display page (alpha modes)
6004                                                                            ; BL = foreground color (graphics modes)
6005 00001ABC B8080E                            mov     ax, 0E08h
6006 00001ABF CD10                              int     10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
6007                                                                            ; AL = character, BH = display page (alpha modes)
6008                                                                            ; BL = foreground color (graphics modes)
6009                                            ;pop    bx
6010                                            ;add    bx, 18                  ; 18.2 ticks per second
6011                                            ; 09/12/2023
6012 00001AC1 83C212                            add     dx, 18
6013 00001AC4 83D600                            adc     si, 0                   ; next second (if carry flag is 1)
6014                                    continue_to_wait:
6015 00001AC7 B401                              mov     ah, 1
6016 00001AC9 CD16                              int     16h                     ; KEYBOARD - status
6017 00001ACB 741B                              jz      short cdbo_5
6018 00001ACD B400                              mov     ah, 0
6019 00001ACF CD16                              int     16h                     ; KEYBOARD - READ CHAR FROM BUFFER, WAIT IF EMPTY
6020                                                                            ; Return: AH = scan code, AL = character
6021
6022                                            ; 09/12/2023
6023                                            ;cmp    ax, 11Bh ; ESC key
6024                                            ;jz     short cdb0_7
6025                                    ;cdbo_4:
6026                                            ;push   ax ; *
6027 00001AD1 89C2                              mov     dx, ax ; *
6028
6029                                            ; CRLF (next line)
6030                                            ;mov    bx, 7
6031                                            ; bx = 7
6032 00001AD3 B80D0E                            mov     ax, 0E0Dh
6033 00001AD6 CD10                              int     10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
6034                                                                            ; AL = character, BH = display page (alpha modes)
6035                                                                            ; BL = foreground color (graphics modes)
6036 00001AD8 B80A0E                            mov     ax, 0E0Ah
6037 00001ADB CD10                              int     10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
6038                                                                            ; AL = character, BH = display page (alpha modes)
6039                                                                            ; BL = foreground color (graphics modes)
6040                                            ; 09/12/2023
6041                                            ;pop    ax ; *
6042
6043 00001ADD 81FA1B01                          cmp     dx, 11Bh
6044                                            ;cmp    ax, 11Bh ; ESC key (to cancel CD/DVD boot)
6045 00001AE1 7418                              je      short cdbo_7
6046
6047                                    cdbo_4:         ; 10/12/2023
6048 00001AE3 5A                                pop     dx
6049 00001AE4 07                                pop     es
6050 00001AE5 1F                                pop     ds
6051 00001AE6 58                                pop     ax
6052 00001AE7 C3                                retn
6053                                    cdbo_5:
6054 00001AE8 3B366E00                          cmp     si, [6Eh]
6055 00001AEC 7504                              jnz     short cdbo_6
6056                                            ; 09/12/2023
6057 00001AEE 3B166C00                          cmp     dx, [6Ch]
6058                                            ;cmp    bx, [6Ch]
6059                                    cdbo_6:
6060 00001AF2 73D3                              jnb     short continue_to_wait
6061 00001AF4 2EFE0E[6C1B]                      dec     byte [cs:time_counter]
6062 00001AF9 75B7                              jnz     short wait_for_key
6063                                    cdbo_7:
6064                                            ; 09/12/2023
6065                                            ; CRLF (next line)
6066                                            ;
6067                                            ;mov    bx, 7
6068                                            ;mov    ax, 0E0Dh
6069                                            ;int    10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
6070                                            ;                               ; AL = character, BH = display page (alpha modes)
6071                                            ;                               ; BL = foreground color (graphics modes)
6072                                            ;mov    ax, 0E0Ah
6073                                            ;int    10h                     ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
6074                                            ;                               ; AL = character, BH = display page (alpha modes)
6075                                            ;                               ; BL = foreground color (graphics modes)
6076
6077 00001AFB 0E                                push    cs
6078 00001AFC 1F                                pop     ds
6079                                            ; 09/12/2023
6080 00001AFD 1E                                push    ds
6081 00001AFE 07                                pop     es
6082                                            ; es = ds = cs
```

```
6083
6084 00001AFF B8004B                    mov     ax, 4B00h
6085                                    ;xor    dl, dl
6086                                    ; 09/12/2023
6087 00001B02 31D2                      xor     dx, dx
6088                                    ; dl = disk drive = 0  ; fd
6089                                    ;mov    si, 1C93h
6090 00001B04 BE[591B]                  mov     si, empty_dap_buff
6091 00001B07 CD13                      int     13h                     ; DISK - Bootable CD-ROM - AL = TERMINATE DISK EMULATION
6092                                    ; DS:SI = Specification packet filled
6093
6094                                    ;mov    dx, 80h
6095                                    ;xor    ax, ax
6096                                    ; 09/12/2023
6097 00001B09 B81300                    mov     ax, 19
6098 00001B0C 89F7                      mov     di, si
6099                                    ;mov    byte [si], 13h
6100                                    ;mov    [si+1], al
6101 00001B0E AB                        stosw
6102                                    ;mov    [si+2], dx
6103 00001B0F B080                      mov     al, 80h
6104 00001B11 AB                        stosw
6105 00001B12 89C2                      mov     dx, ax
6106                                    ;mov    [si+4], ax
6107                                    ;mov    [si+6], ax
6108                                    ;mov    [si+8], ax
6109                                    ;mov    [si+0Ah], ax
6110                                    ;mov    [si+0Ch], ax
6111                                    ;mov    [si+0Eh], ax
6112                                    ;mov    [si+10h], al
6113                                    ;mov    [si+11h], al
6114                                    ;mov    [si+12h], al
6115 00001B14 B90F00                    mov     cx, 15
6116 00001B17 F3AA                      rep     stosb
6117                                    ; dl = disk drive = 80h ; hd
6118 00001B19 B8004B                    mov     ax, 4B00h
6119 00001B1C CD13                      int     13h                     ; DISK - Bootable CD-ROM - AL = TERMINATE DISK EMULATION
6120 00001B1E 31C0                      xor     ax, ax
6121                                    ; 09/12/2023
6122                                    ;mov    dx, 80h
6123                                    ; dx = 80h
6124 00001B20 CD13                      int     13h                     ; DISK - RESET DISK SYSTEM
6125                                                                    ; DL = drive (if bit 7 is set both hard disks and floppy disks reset)
6126                                    ; 09/12/2023
6127                                    ;push   cs
6128                                    ;pop    es
6129                                    ; es = ds = cs
6130
6131 00001B22 B80102                    mov     ax, 201h
6132                                    ;mov    bx, 152h
6133 00001B25 BB[5201]                  mov     bx, disksector
6134                                    ;mov    cx, 1
6135                                    ; 09/12/2023
6136 00001B28 41                        inc     cx ; cx = 1
6137                                    ;mov    dx, 80h
6138                                    ; dx = 80h
6139 00001B29 CD13                      int     13h                     ; DISK - READ SECTORS INTO MEMORY
6140                                                                    ; AL = number of sectors to read, CH = track, CL = sector
6141                                                                    ; DH = head, DL = drive, ES:BX -> buffer to fill
6142                                                                    ; Return: CF set on error, AH = status, AL = number of sectors read
6143                                    ;jc     short cdbo_8
6144                                    ; 10/12/2023
6145 00001B2B 72B6                      jc      short cdbo_4
6146
6147 00001B2D 2681BFFE0155AA            cmp     word [es:bx+1FEh], 0AA55h
6148                                    ;jz     short cdbo_9
6149                                    ; 10/12/2023
6150 00001B34 75AD                      jnz     short cdbo_4
6151                                ;cdbo_8:
6152                                    ;jmp    short cdbo_4
6153                                ;cdbo_9:
6154                                    ; 10/12/2023
6155                                    ; (stack clearing -pop- is not necessary here,
6156                                    ;  PCDOS 7.1 boot sector will set stack pointer again)
6157                                    ;pop    ax ; near call return address
6158                                    ;pop    cx ; +++ ; ch = [MediaByte]
6159
6160                                    ; 09/12/2023
6161                                    ;push   cs
6162                                    ;pop    ds
6163                                    ; ds = cs
6164 00001B36 31C0                      xor     ax, ax  ; 0
6165 00001B38 BF007C                    mov     di, 7C00h
6166 00001B3B 8EC0                      mov     es, ax
6167 00001B3D 89DE                      mov     si, bx
6168 00001B3F 06                        push    es
6169 00001B40 57                        push    di
6170 00001B41 B90001                    mov     cx, 100h ; 256
6171                                    ; 10/12/2023
6172                                    ;cld    ; not necessary (direction flag is already cleared)
6173 00001B44 F3A5                      rep movsw
6174 00001B46 8ED8                      mov     ds, ax
6175 00001B48 BE7800                    mov     si, 78h
6176 00001B4B 2EA1[141A]                mov     ax, [cs:Orig_Int1Eh_Table]
6177 00001B4F 8904                      mov     [si], ax
6178 00001B51 2EA1[161A]                mov     ax, [cs:Orig_Int1Eh_Table+2]
6179 00001B55 894402                    mov     [si+2], ax
6180 00001B58 CB                        retf
6181
6182                                ; ---------------------------------------------------------------------------
6183                                dap_buffer: ; 16/12/2023
6184
6185 00001B59 13                    empty_dap_buff:     db 19
6186                                                    ;db 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0  ; db 18 dup(0)
6187 00001B5A 00<rep 12h>                               times 18 db 0
6188 00001B6C 05                    time_counter:       db 5    ; 5 seconds
6189 00001B6D 0D0A                  cd_boot_msg:        db 0Dh,0Ah
6190                                                    ;db 'Press the ENTER key to boot from CD or DVD......',0
6191                                                    ; 09/12/2023
6192 00001B6F 507265737320616E79-                       db 'Press any key to boot from CD or DVD ...',0
6192 00001B78 206B657920746F2062-
6192 00001B81 6F6F742066726F6D20-
6192 00001B8A 4344206F7220445644-
6192 00001B93 202E2E2E00
6193
6194                                ; ---------------------------------------------------------------------------
6195
6196                                ; 01/10/2022 - Retro DOS v4.0 (MSDOS 5.0)
6197
6198                                ; 10/12/2023 - Retro DOS v5.0 (PCDOS 7.1)
6199                                ; (PCDOS 7.1 IBMBIO.COM, BIOSDATA:1CDAh)
6200
6201                                ;---------------------------------------------------------------------------
6202                                ; entry point from boot sector
```

```
                        ;--------------------------------------------------------------------------

                        init:               ; 27/12/2018
                                    ; 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
00001B98 1E                         push    ds
00001B99 07                         pop     es
00001B9A 0E                         push    cs
00001B9B 1F                         pop     ds
00001B9C 8C06[161A]                 mov     [Orig_Int1Eh_Table+2], es
00001BA0 8936[141A]                 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
00001BA4 A3[001A]                       mov     [First_Data_Sector+2], ax
00001BA7 891E[FE19]                     mov     [First_Data_Sector], bx
00001BAB 8816[821A]                     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.

00001BAF 0E                             push    cs              ; Save a peck of interrupt vectors...
00001BB0 07                             pop     es

00001BB1 51                             push    cx ; +++ ; ch = [MediaByte]
                                        ;push   di ; *! (not necessary) ; 10/12/2023

                                        ; 10/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
00001BB2 FC                             cld     ; (may not be necessary)

00001BB3 31C0                           xor     ax, ax
00001BB5 8ED8                           mov     ds, ax  ; ds = 0

                                        ; 06/04/2024
00001BB7 50                             push    ax ; push ds ; 0

                                        ;mov    ax, 544h        ; SYSINIT segment
00001BB8 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)

00001BBB BE4005                         mov     si, 540h  ; ROOT DIRECTORY BUFFER offset 40h
                                                          ; (BOOT DRV's root directory the 3rd entry)
                                chk_boot_hdnoz:
00001BBE 803C00                         cmp     byte [si], 0
00001BC1 7436                           jz      short chk_no_logo_noz
00001BC3 813C5F42                       cmp     word [si], 425Fh ; '_BOOT_HDNOZ'
00001BC7 7527                           jnz     short chk_next_1
00001BC9 817C024F4F                     cmp     word [si+2], 4F4Fh ; 'OO'
00001BCE 7520                           jnz     short chk_next_1
00001BD0 817C04545F                     cmp     word [si+4], 5F54h
00001BD5 7519                           jnz     short chk_next_1
00001BD7 817C064844                     cmp     word [si+6], 4448h ; 'HD'
00001BDC 7512                           jnz     short chk_next_1
00001BDE 817C084E4F                     cmp     word [si+8], 4F4Eh
```

```
6327 00001BE3 750B                      jnz     short chk_next_1
6328 00001BE5 807C0A5A                  cmp     byte [si+0Ah], 5Ah ; 'Z'
6329 00001BE9 7505                      jnz     short chk_next_1
6330 00001BEB E895FE                    call    cd_boot_option
6331 00001BEE EB09                      jmp     short chk_no_logo_noz
6332                                 chk_next_1:
6333 00001BF0 83C620                    add     si, 32  ; (next entry)
6334 00001BF3 81FE0007                  cmp     si, 700h
6335 00001BF7 72C5                      jb      short chk_boot_hdnoz
6336                                 chk_no_logo_noz:
6337 00001BF9 BE4005                    mov     si, 540h  ; (BOOT DRV's root directory the 3rd entry)
6338                                 chk_no_logo_noz2_nxt:
6339 00001BFC 803C00                    cmp     byte [si], 0
6340 00001BFF 7431                      jz      short write_start_msg
6341 00001C01 813C4E4F                  cmp     word [si], 4F4Eh ; 'NO_LOGO NOZ'
6342 00001C05 7522                      jnz     short chk_next_2
6343 00001C07 817C025F4C                cmp     word [si+2], 4C5Fh
6344 00001C0C 751B                      jnz     short chk_next_2
6345 00001C0E 817C044F47                cmp     word [si+4], 474Fh
6346 00001C13 7514                      jnz     short chk_next_2
6347 00001C15 817C064F20                cmp     word [si+6], 204Fh
6348 00001C1A 750D                      jnz     short chk_next_2
6349 00001C1C 817C084E4F                cmp     word [si+8], 4F4Eh
6350 00001C21 7506                      jnz     short chk_next_2
6351 00001C23 807C0A5A                  cmp     byte [si+0Ah], 5Ah
6352 00001C27 741C                      jz      short startmsg_ok
6353                                 chk_next_2:
6354 00001C29 83C620                    add     si, 32  ; (next entry)
6355 00001C2C 81FE0007                  cmp     si, 700h
6356 00001C30 72CA                      jb      short chk_no_logo_noz2_nxt
6357                                 write_start_msg:
6358 00001C32 8ED8                      mov     ds, ax  ; SYSINIT segment
6359 00001C34 BE[C751]                  mov     si, StartMsg   ; "Starting PC DOS...\r\n\n"
6360                                 startmsg_nxt_chr:
6361 00001C37 AC                        lodsb
6362 00001C38 08C0                      or      al, al
6363 00001C3A 7409                      jz      short startmsg_ok
6364 00001C3C B40E                      mov     ah, 0Eh
6365 00001C3E BB0700                    mov     bx, 7
6366 00001C41 CD10                      int     10h             ; - VIDEO - WRITE CHARACTER AND ADVANCE CURSOR (TTY WRITE)
6367                                                             ; AL = character, BH = display page (alpha modes)
6368                                                             ; BL = foreground color (graphics modes)
6369 00001C43 EBF2                      jmp     short startmsg_nxt_chr
6370                                 startmsg_ok:
6371                                     ; 06/04/2024
6372 00001C45 1F                        pop     ds ; 0
6373
6374                                     ; 10/12/2023
6375                                     ; ds = 0
6376
6377                                     ; 21/12/2022
6378                                     ; ds = 0 (?)
6379                                     ; 24/12/2022
6380                                     ; ds = cs
6381                                     ;xor    cx, cx
6382                                     ;mov    ds, cx
6383                                     ; ds = 0
6384
6385                                     ;mov    cl, 5
6386                                     ; 10/12/2023
6387 00001C46 B90500                    mov     cx, 5           ; NUMROMVECTORS
6388                                                             ; no. of rom vectors to      be saved
6389                                     ;mov    si, offset RomVectors ; point to list of int vectors
6390 00001C49 BE[0001]                  mov     si, RomVectors
6391
6392                                     ; 10/12/2023
6393 00001C4C FA                        cli
6394                                 next_int_:
6395 00001C4D 2E                        cs      ; 16/10/2022
6396 00001C4E AC                        lodsb
6397                                     ;lods   byte ptr cs:[si] ; cs lodsb
6398 00001C4F 98                        cbw                     ; ax = interrupt number
6399 00001C50 D1E0                      shl     ax, 1
6400 00001C52 D1E0                      shl     ax, 1           ; int no * 4
6401 00001C54 89C7                      mov     di, ax          ; interrupt vector address
6402 00001C56 87FE                      xchg    si, di          ; rombios interrupt vector address in si
6403                                                             ; saving address in di
6404                                     ;lodsw                  ; movsw
6405                                     ;stosw
6406                                     ;lodsw                  ; movsw
6407                                     ;stosw                  ; save the vector
6408                                     ; 21/04/2024
6409 00001C58 A5                        movsw
6410 00001C59 A5                        movsw
6411
6412 00001C5A 87FE                      xchg    si, di
6413 00001C5C E2EF                      loop    next_int_
6414
6415                                     ; 10/12/2023
6416                                     ;pop    di  ; *!
6417                                     ;pop    cx ; +++ ; ch = [MediaByte]
6418
6419                                 ; we need to save int13 in two places in case we are running on an at.
6420                                 ; on ats we install the ibm supplied rom_bios patch which hooks
6421                                 ; int13 ahead of orig13. since int19 must unhook int13 to point to the
6422                                 ; rom int13 routine, we must have that rom address also stored away.
6423
6424                                     ; 21/12/2022
6425                                     ;mov    ax, [cs:Old13] ; save old13 in orig13 also
6426                                     ;mov    [cs:Orig13], ax
6427                                     ;mov    ax, [cs:Old13+2]
6428                                     ;mov    [cs:Orig13+2], ax
6429
6430                                     ; 16/10/2022
6431 00001C5E C7064C00[ED06]              mov     word [13h*4], block13
6432                                     ;mov    word ptr ds:4Ch, offset      block13 ; 13h*4
6433                                                             ; set up int 13       for new action
6434 00001C64 8C0E4E00                  mov     [13h*4+2], cs
6435                                     ;mov    word ptr ds:4Eh, cs ; 13h*4+2
6436 00001C68 C7065400[9907]            mov     word [15h*4], Int15
6437                                     ;mov    word ptr ds:54h, offset      Int15 ; 15h*4
6438                                                             ; set up int 15       for new action
6439 00001C6E 8C0E5600                  mov     [15h*4+2], cs
6440                                     ;mov    word ptr ds:56h, cs ; 15h*4+2
6441 00001C72 C7066400[5907]            mov     word [19h*4], int19
6442                                     ;mov    word ptr ds:64h, offset      int19 ; 19h*4
6443                                                             ; set up int 19       for new action
6444 00001C78 8C0E6600                  mov     [19h*4+2], cs
6445                                     ;mov    word ptr ds:66h, cs ; 19h*4+2
6446
6447                                     ; 10/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
6448 00001C7C A16800                    mov     ax, [68h]       ; 1Ah*4
6449 00001C7F 8B3E6A00                  mov     di, [6Ah]       ; 1Ah*4+2
6450 00001C83 C7066800[AF06]            mov     word [68h], Int1A
```

```
00001C89 8C0E6A00                       mov     [6Ah], cs

                                        ; 21/12/2022
00001C8D 0E                             push    cs
00001C8E 1F                             pop     ds

                                        ; 10/12/2023
00001C8F A3[AB06]                       mov     [Orig1A], ax
00001C92 893E[AD06]                     mov     [Orig1A+2], di

00001C96 A1[0601]                       mov     ax, [Old13]     ; save old13 in orig13 also
00001C99 A3[B400]                       mov     [Orig13], ax
00001C9C A1[0801]                       mov     ax, [Old13+2]
00001C9F A3[B600]                       mov     [Orig13+2], ax
                                                ;
00001CA2 FB                             sti
00001CA3 CD11                           int     11h             ; EQUIPMENT DETERMINATION
                                                ; Return: AX = equipment flag bits
                                        ; 10/12/2023
                                        ;jmp    short chk_fd_count
                                        ; (PCDOS 7.1 IBMBIO.COM, BIOSDATA:1DF7h) ; *!!*
                                        ; ((signature))
                                        ;push   dx              ; 52h ; 'R'
                                        ;push   ax              ; 50h ; 'P'
                                        ;push   bx              ; 53h ; 'S'

                                ; we have to support a system that does not have any diskette
                                ; drives but only hardfiles. this system will ipl from the hardfile.
                                ; if the equipment flag bit 0 is 1, then the system has diskette drive(s).
                                ; otherwise, the system has only hardfiles.
                                ;
                                ; important thing is that still, for compatibility reason, the drive letter
                                ; for the hardfiles start from "c". so, we still need to allocate dummy bds
                                ; drive a and drive b. at sysinit time, we are going to set cds table entry
                                ; of dpb pointer for these drives to 0, so any user attempt to access this
                                ; drives will get "invalid drive letter ..." message. we are going to
                                ; establish "fakefloppydrv" flag. ***sysinit module should call int 11h to
                                ; determine whether there are any diskette drivers in the system or not.!!!***

                                ; check the register returned by the equipment determination interrupt
                                ; we have to handle the case of no diskettes in the system by faking
                                ; two dummy drives.
                                ;
                                ; if the register indicates that we do have floppy drives we don't need
                                ; to do anything special.
                                ;
                                ; if the register indicates that we don't have any floppy drives then
                                ; what we need to do is set the fakefloppydrv variable, change the
                                ; register to say that we do have floppy drives and then go to execute
                                ; the code which starts at notsingle. this is because we can skip the
                                ; code given below which tries to find if there are one or two drives
                                ; since we already know about this.

                                chk_fd_count:   ; 10/12/2023
                                        ;or     ax, 1   ; *!!*

                                        ; 12/12/2022
00001CA5 A801                           test    al, 1
                                        ;test   ax, 1           ; floppy drives       present?
00001CA7 7525                           jnz     short normalfloppydrv ;       yes.

                                ; Some ROM BIOSs lie that there are no floppy drives. Lets find out
                                ; whether it is an old ROM BIOS or a new one
                                ;
                                ; WARNING !!!
                                ;
                                ; This sequence of code is present in SYSINIT1.ASM also. Any modification
                                ; here will require an equivalent modification in SYSINIT1.ASM also

                                        ; 10/12/2023
                                        ; ((cx is already on top of the stack))
                                        ;push   cx      ; +++ ; ch = [MediaByte]
                                        ;push   bx      ; not necessary
00001CA9 50                             push    ax
00001CAA 52                             push    dx
                                        ;push   di      ; not necessary
00001CAB 06                             push    es

00001CAC B408                           mov     ah, 8
00001CAE B200                           mov     dl, 0
00001CB0 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
00001CB2 7204                           jc      short _gdskp_error
                                        ;mov    [cs:flp_drvs], dl
                                        ; 21/12/2022
                                        ; ds = cs
00001CB4 8816[7F1A]                     mov     [flp_drvs], dl
                                _gdskp_error:
                                        ; 10/12/2023
00001CB8 07                             pop     es
                                        ;pop    di
00001CB9 5A                             pop     dx
00001CBA 58                             pop     ax
                                        ;pop    bx
00001CBB 59                             pop     cx      ; +++ ; ch = [MediaByte]
00001CBC 7210                           jc      short normalfloppydrv
                                                        ; if error it is an old ROM BIOS
                                                        ; so, lets assume that ROM BIOS lied
                                        ; 21/12/2022
                                        ;cmp    byte [cs:flp_drvs], 0 ; number of drvs == 0?
                                        ;jz     short _set_fake_flpdrv
                                        ;mov    al, [cs:flp_drvs]
00001CBE A0[7F1A]                       mov     al, [flp_drvs]
00001CC1 08C0                           or      al, al          ; number of drvs == 0?
00001CC3 7403                           jz      short _set_fake_flpdrv

                                        ;dec    al              ; make it zero based
                                        ; 18/12/2022
00001CC5 48                             dec     ax
00001CC6 EB0A                           jmp     short got_num_flp_drvs
                                ; ---------------------------------------------------------------------------

                                _set_fake_flpdrv:
                                        ; 21/12/2022
                                        ;mov    ax, 1
                                        ; 10/12/2023
00001CC8 40                             inc     ax ; al = 1
00001CC9 A2[131A]                       mov     [fakefloppydrv], al ; 1
                                        ;mov    byte [cs:fakefloppydrv], 1
                                        ;                       ; we don't have any floppy drives.
                                        ;mov    ax, 1
```

```
6575 00001CCC EB09                      jmp     short settwodrive ; well then set it for two drives!
6576                                    ; ---------------------------------------------------------------------------
6577
6578                                    normalfloppydrv:                ; yes, bit 0 is 1.
6579 00001CCE D0C0                      rol     al, 1           ; there exist floppy drives.
6580 00001CD0 D0C0                      rol     al, 1           ; put bits 6 & 7 into bits 0 & 1
6581                                    got_num_flp_drvs:
6582                                    ;and    ax, 3           ; only look at bits 0 &       1
6583                                    ; 18/12/2022
6584 00001CD2 2403                      and     al, 3
6585 00001CD4 7505                      jnz     short notsingle         ; zero means single drive system
6586 00001CD6 40                        inc     ax              ; pretend it's a two drive system
6587                                    settwodrive:                            ; set this to two fakedrives
6588                                    ; 21/12/2022
6589                                    ; ds = cs
6590 00001CD7 FE06[7800]                inc     byte [single]
6591                                    ;inc    byte [cs:single] ; remember this
6592                                    notsingle:
6593 00001CDB 40                        inc     ax              ; ax has number         of drives, 2-4
6594                                                            ; is also 0 indexed boot drive if we
6595                                                            ; booted off hard file
6596 00001CDC 88C1                      mov     cl, al          ; ch is fat id, cl # floppies
6597
6598                                    ; 16/10/2022
6599                                    ; MSDOS 3.3 - "MSEQU.INC" (24/07/1987)
6600                                    INITSPOT EQU        534h    ; IBM wants 4 zeros here
6601                                    BRKADR      EQU     1BH * 4 ; 6CH, 1BH break vector address
6602                                    TIMADR      EQU     1CH * 4 ; 70H, 1CH timer interrupt
6603                                    DSKADR      EQU     1EH * 4 ; address of ptr to disk parameters
6604                                    SEC9EQU     522h    ; address of disk parameters
6605                                    CHROUT      EQU     29h
6606                                    LSTDRV      EQU      504h
6607
6608                                    ; determine whether we booted from floppy or hard disk...
6609
6610 00001CDE F6C280                    test    dl, 80h         ; boot from floppy ?
6611 00001CE1 7505                      jnz     short gothrd    ; no.
6612 00001CE3 31C0                      xor     ax, ax          ; indicate boot         from drive a
6613                                                            ; 10/12/2023
6614 00001CE5 A2[821A]                  mov      [Boot_Drv], al
6615                                    gothrd:
6616 00001CE8 31D2                      xor     dx, dx ; 0      ; ax = 0-based drive we         booted from
6617                                                            ; bios_l, bios_h set.
6618                                                            ; cl = number of floppies including fake one
6619                                                            ; ch = media byte
6620 00001CEA FA                        cli
6621 00001CEB 8ED2                      mov     ss, dx          ; set stack segment and stack pointer
6622 00001CED BC0007                    mov     sp, 700h
6623 00001CF0 FB                        sti
6624
6625 00001CF1 51                        push    cx ; *          ; save number of floppies and media byte
6626 00001CF2 88EC                      mov     ah, ch          ; FAT ID to AH
6627 00001CF4 50                        push    ax ; **         ; save boot drive number and media byte
6628
6629                                    ; let model_byte, secondary_model_byte be set here!!!
6630
6631 00001CF5 B4C0                      mov     ah, 0C0h
6632 00001CF7 CD15                      int     15h             ; SYSTEM - GET CONFIGURATION (XT after 1/10/86,AT mdl 3x9,CONV,XT286,PS)
6633 00001CF9 7215                      jb      short no_rom_system_conf ; just       use Model_Byte
6634 00001CFB 80FC00                    cmp     ah, 0
6635 00001CFE 7510                      jnz     short no_rom_system_conf
6636
6637                                                            ; 21/12/2022
6638                                                            ; ds = cs
6639 00001D00 268A4702                  mov     al, [es:bx+2]  ; [es:bx+ROMBIOS_DESC.bios_sd_modelbyte]
6640 00001D04 A2[AF05]                  mov     [model_byte], al
6641                                                            ;mov    [cs:model_byte], al
6642                                                                    ; get/save model byte
6643 00001D07 268A4703                  mov     al, [es:bx+3]  ; [es:bx+ROMBIOS_DESC.bios_sd_scnd_modelbyte]
6644 00001D0B A2[B005]                  mov     [secondary_model_byte], al
6645                                                            ;mov    [cs:secondary_model_byte], al
6646                                                                    ; get/save secondary model byte
6647 00001D0E EB0C                      jmp     short turn_timer_on
6648                                    ;---------------------------------------------------------------------------
6649
6650                                    no_rom_system_conf:
6651 00001D10 BEFFFF                    mov     si, 0FFFFh
6652 00001D13 8EC6                      mov     es, si
6653                                    ; 21/12/2022
6654 00001D15 26A00E00                  mov     al, [es:0Eh]   ; get model byte (from 0FFFFh:0Eh)
6655 00001D19 A2[AF05]                  mov     [model_byte], al
6656                                    ;mov    [cs:model_byte], al ; save model byte
6657                                    turn_timer_on:
6658 00001D1C B020                      mov     al, 20h ; ' '   ; turn on the timer
6659 00001D1E E620                      out     20h, al         ; Interrupt controller,         8259A.
6660                                                            ; AKPORT
6661
6662                                    ; some olivetti m24 machines have an 8530 serial communications
6663                                    ; chip installed at io address 50h and 52h. if we're running
6664                                    ; on one of those, we must inhibit the normal aux port initialization
6665
6666                                                    ; 21/12/2022
6667                                                    ; ds = cs
6668 00001D20 803E[AF05]00                      cmp     byte [model_byte], 0
6669                                            ;cmp    byte [cs:model_byte], 0 ; next to last          byte in rom bios
6670 00001D25 7510                              jnz     short not_olivetti_m24 ; skip for all other machines
6671                                                                    ; (except olivetti m24)
6672 00001D27 E466                              in      al, 66h         ; is 8530 installed?
6673 00001D29 A820                              test    al, 20h
6674 00001D2B 740A                              jz      short not_olivetti_m24 ; we're done if not
6675 00001D2D B00F                              mov     al, 0Fh         ; double check
6676 00001D2F E650                              out     50h, al
6677 00001D31 E450                              in      al, 50h
6678 00001D33 A801                              test    al, 1           ; this test was         copied from olivetti
6679 00001D35 7414                              jz      short skip_aux_port_init ; take         this branch if 8530 installed
6680
6681                                    not_olivetti_m24:
6682 00001D37 B003                              mov     al, 3           ; init com4
6683 00001D39 E8F00A                            call    aux_init
6684 00001D3C B002                              mov     al, 2           ; init com3
6685 00001D3E E8EB0A                            call    aux_init
6686 00001D41 B001                              mov     al, 1           ; init com2
6687 00001D43 E8E60A                            call    aux_init
6688 00001D46 30C0                              xor     al, al          ; init com1
6689 00001D48 E8E10A                            call    aux_init
6690
6691                                    skip_aux_port_init:
6692 00001D4B B002                              mov     al, 2           ; init lpt3
6693 00001D4D E8D40A                            call    print_init
6694 00001D50 B001                              mov     al, 1           ; init lpt2
6695 00001D52 E8CF0A                            call    print_init
6696 00001D55 30C0                              xor     al, al          ; init lpt1
6697 00001D57 E8CA0A                            call    print_init
6698
```

```
6699 00001D5A 31D2                      xor     dx, dx ; 0
6700 00001D5C 8EDA                      mov     ds, dx          ; to initialize       print screen vector
6701 00001D5E 8EC2                      mov     es, dx
6702
6703                                    ; 11/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
6704 00001D60 BF3405                    mov     di, 534h        ; offset INITSPOT
6705                                    ;mov    di, INITSPOT    ; 0534h
6706                                                            ; IBMDOS.COM's first cluster - high word
6707                                                            ; 520h (the 2nd entry of root dir) + 14h
6708 00001D63 8B05                      mov     ax, [di]
6709                                    ;mov    [firstcluster_hw], ax
6710                                    ; 06/04/2024
6711 00001D65 2EA3[801A]                mov     [cs:firstcluster_hw], ax
6712
6713 00001D69 31C0                      xor     ax, ax
6714                                    ; 11/12/2023
6715                                    ; 16/10/2022
6716                                    ;mov    di, INITSPOT    ; 0534h
6717                                    ;;mov   di, 534h        ; INITSPOT (0000h:0534h)
6718                                                            ; IBM wants 4 zeros here
6719 00001D6B AB                        stosw
6720 00001D6C AB                        stosw
6721 00001D6D 8CC8                      mov     ax, cs          ; fetch segment
6722 00001D6F C7066C00[0E06]            mov     word [BRKADR], cbreak
6723                                    ;mov    word ptr ds:6Ch, offset       cbreak ; [BRKADR]
6724                                                            ; break entry point
6725 00001D75 A36E00                    mov     [BRKADR+2], ax
6726                                    ;mov    ds:6Eh, ax      ; vector for break
6727 00001D78 C706A400[8206]            mov     word [CHROUT*4], outchr
6728                                    ;mov    word ptr ds:0A4h, offset outchr       ; [CHROUT*4]
6729 00001D7E A3A600                    mov     [CHROUT*4+2], ax
6730                                    ;mov    ds:0A6h, ax     ; [CHROUT*4+2]
6731 00001D81 BF0400                    mov     di, 4
6732 00001D84 BB[1406]                  mov     bx, intret ; 19/10/2022
6733                                    ;mov    bx, offset intret ; intret (cs:intret)
6734                                                            ; will initialize rest of interrupts
6735 00001D87 93                        xchg    ax, bx
6736 00001D88 AB                        stosw                   ; location 4
6737 00001D89 93                        xchg    ax, bx          ; cs:
6738 00001D8A AB                        stosw                   ; int 1 ; location 6
6739 00001D8B 83C704                    add     di, 4           ; skip int 2
6740 00001D8E 93                        xchg    ax, bx
6741 00001D8F AB                        stosw                   ; location 12
6742 00001D90 93                        xchg    ax, bx          ; cs:
6743 00001D91 AB                        stosw                   ; int 3 ; location 14
6744 00001D92 93                        xchg    ax, bx
6745 00001D93 AB                        stosw                   ; location 16
6746 00001D94 93                        xchg    ax, bx          ; cs:
6747 00001D95 AB                        stosw                   ; int 4 ; location 18
6748 00001D96 89160005                  mov     [0500h], dx
6749                                    ;mov    ds:500h, dx     ; set print screen & break = 0
6750 00001D9A 89160405                  mov     [LSTDRV], dx    ; [0504h]
6751                                    ;mov    ds:504h, dx     ; clean out last drive spec
6752
6753                                ; we need to initalize the cs:motorstartup variable from the disk
6754                                ; parameter table at sec9. the offsets in this table are defined in
6755                                ; the disk_parms struc in msdskprm.inc. 2 locs
6756
6757 00001D9E A02C05                        mov     al, [SEC9+0Ah] ; 16/10/2022
6758                                        ;mov    al, ds:52Ch     ; [SEC9+DISK_PARMS.DISK_MOTOR_STRT]
6759                                                                ; [522h+0Ah]
6760                                        ; 21/12/2022
6761                                        ; ds = 0
6762
6763 00001DA1 2EA2[2601]                    mov     [cs:motorstartup], al
6764 00001DA5 2E803E[AF05]FD                cmp     byte [cs:model_byte], 0FDh ; is this an old rom?
6765 00001DAB 720B                          jb      short no_diddle         ; no
6766 00001DAD C7062B050F02                  mov     word [SEC9+09h], 20Fh
6767                                        ;mov    word ptr ds:52Bh, 20Fh ; [SEC9+DISK_PARMS.DISK_HEAD_STTL], 0200h+NORMSETTLE
6768                                                                ; set head settle and motor start on pc-1 pc-2 pc-xt hal0
6769 00001DB3 C6062205DF                   mov     byte [SEC9+0], 0DFh
6770                                        ;mov    byte ptr ds:522h, 0DFh ; [SEC9+DISK_PARMS.DISK_SPECIFY_1]
6771                                                                ;  set 1st specify byte       on pc-1 pc-2 pc-xt hal0
6772                                no_diddle:
6773 00001DB8 CD12                         int     12h             ; MEMORY SIZE -
6774                                                                ; Return: AX = number of contiguous 1K blocks of memory
6775 00001DBA B106                         mov     cl, 6
6776 00001DBC D3E0                         shl     ax, cl          ; convert memory size to 16-byte blocks     (segment no.)
6777
6778                                        ; 21/12/2022
6779                                        ;pop    cx
6780                                        ;mov    [cs:drvfat], cx ; save drive to load dos, and fat id
6781
6782 00001DBE 50                           push    ax ; ***        ; save real top       of memory
6783
6784                                ;M068 - BEGIN
6785                                ;------ Check if an RPL program is present at TOM and do not tromp over it
6786
6787                                        ; 21/12/2022
6788                                        ; ds = 0
6789                                        ;push   ds
6790                                        ;push   bx              ; pushes not required but since this
6791                                        ;                       ; happens to be a last minute change
6792                                        ;                       ; & since it is only init code.
6793                                        ;xor    bx, bx
6794                                        ;mov    ds, bx
6795
6796                                        ;mov    bx, ds:0BCh     ; [2Fh*4]
6797 00001DBF 8B1EBC00                     mov     bx, [2Fh*4]
6798                                        ;mov    ds, word ptr ds:0BEh ; [2Fh*4+2]
6799 00001DC3 8E1EBE00                     mov     ds, [2Fh*4+2]
6800 00001DC7 817F035250                   cmp     word [bx+3], 'RP' ; 'RPL'
6801                                        ;cmp    word ptr [bx+3], 'PR' ; 'RPL'
6802 00001DCC 750F                         jnz     short SkipRPL
6803 00001DCE 807F054C                     cmp     byte [bx+5], 'L'
6804                                        ;cmp    byte ptr [bx+5], 'L'
6805 00001DD2 7509                         jnz     short SkipRPL
6806 00001DD4 89C2                         mov     dx, ax          ; get TOM into DX
6807 00001DD6 B8064A                       mov     ax, 4A06h       ; (multMULT shl       8) + multMULTRPLTOM
6808 00001DD9 CD2F                         int     2Fh             ; Get new TOM from any RPL
6809 00001DDB 89D0                         mov     ax, dx
6810                                SkipRPL:
6811                                        ; 21/12/2022
6812                                        ;pop    bx
6813                                        ;pop    ds
6814                                ;M068 - END
6815                                        ; 21/12/2022
6816 00001DDD 0E                           push    cs
6817 00001DDE 1F                           pop     ds
6818
6819 00001DDF 83E840                       sub     ax, 64          ; room for fatloc segment. (1 kb buffer)
6820                                        ; 21/12/2022
6821 00001DE2 A3[041A]                     mov     [fatloc], ax
6822                                        ;mov    [cs:fatloc], ax       ; location to read fat
```

```
6823
6824 00001DE5 83E840                          sub     ax, 64
6825 00001DE8 A3[061A]                        mov     [init_bootseg], ax ; 21/12/2022
6826                                          ;mov    [cs:init_bootseg], ax
6827 00001DEB 58                              pop     ax ; ***        ; get back real top of memory for
6828
6829                                          ; 21/12/2022
6830 00001DEC 59                              pop     cx ; **
6831 00001DED 890E[FC19]                      mov     [drvfat], cx    ; save drive to load dos, and fat id
6832
6833                                          ;;mov   dx, 46Dh        ; SYSINIT segment
6834                                          ;mov    dx, 544h        ; 10/12/2023 (PCDOS 7.1 IBMBIO.COM)
6835 00001DF1 BA0405                          mov     dx, SYSINITSEG ; 17/10/2022
6836 00001DF4 8EDA                            mov     ds, dx
6837
6838                                  ; set pointer to resident device driver chain
6839
6840                                          ; 17/10/2022
6841 00001DF6 C706[7502][2300]                mov     word [DEVICELIST], res_dev_list
6842                                          ;mov    word [273h], res_dev_list
6843                                          ;;mov   word ptr ds:273h, offset res_dev_list
6844                                                                  ; [SYSINIT+DEVICE_LIST]
6845 00001DFC 8C0E[7702]                      mov     [DEVICELIST+2], cs
6846                                          ;mov    [275h], cs
6847                                          ;;mov   word ptr ds:275h, cs ; [SYSINIT+DEVICE_LIST+2]
6848
6849 00001E00 A3[9402]                        mov     [MEMORYSIZE], ax
6850                                          ;mov    [292h], ax
6851                                          ;;mov   ds:292h, ax     ; [SYSINIT+MEMORY_SIZE]
6852
6853 00001E03 FEC1                            inc     cl
6854 00001E05 880E[9802]                      mov     [DEFAULTDRIVE], cl
6855                                          ;mov    [296h], cl
6856                                          ;;mov   ds:296h, cl     ; [SYSINIT+DEFAULT_DRIVE]
6857
6858                                          ;mov    word [CURRENTDOSLOCATION], 0AF8h ; 10/12/2023
6859 00001E09 C706[7302]450A                  mov     word [CURRENTDOSLOCATION], DOSLOADSEG
6860                                          ;mov    word [271h], 83Fh ; (MSDOS.SYS segment)
6861                                          ;;mov   word ptr ds:271h, 83Fh ; [SYSINIT+CURRENT_DOS_LOCATION]
6862                                                                  ; dos_load_seg
6863
6864                                  ; important: some old ibm hardware generates spurious int 0F's due to bogus
6865                                  ; printer cards. we initialize this value to point to an iret only if
6866                                  ;
6867                                  ; 1) the original segment points to storage inside valid ram.
6868                                  ;
6869                                  ; 2) the original segment is 0F000:xxxx
6870
6871                                          ;;;mov  ax, 46Dh        ; SYSINIT segment
6872                                          ;;mov   ax, 544h        ; 10/12/2023
6873                                          ;mov    ax, SYSINITSEG ; 17/10/2022
6874                                          ;mov    es, ax
6875                                          ; 21/12/2022
6876 00001E0F 8EC2                            mov     es, dx ; SYSINITSEG
6877 00001E11 31C9                            xor     cx, cx ; 0
6878 00001E13 8ED9                            mov     ds, cx          ; segment 0
6879                                          ;mov    ax, ds:3Eh      ; [0Fh*4+2]
6880 00001E15 A13E00                          mov     ax, [0Fh*4+2]   ; segment for INT 0Fh
6881                                          ; 18/10/2022
6882 00001E18 263B06[9402]                    cmp     ax, [es:MEMORYSIZE] ; es:292h
6883                                          ;cmp    ax, es:292h     ; [ES:SYSINIT+MEMORY_SIZE]  ; (condition 1)
6884 00001E1D 7605                            jbe     short resetintf
6885 00001E1F 3D00F0                          cmp     ax, 0F000h      ; (condition 2)
6886 00001E22 750A                            jnz     short keepintf
6887                                  resetintf:
6888 00001E24 C7063C00[1406]                  mov     word [0Fh*4], intret
6889                                          ;mov    word ptr ds:3Ch, offset       intret ; [0Fh*4]
6890 00001E2A 8C0E3E00                        mov     word [0Fh*4+2], cs
6891                                          ;mov    word ptr ds:3Eh, cs ; [0Fh*4+2]
6892                                  keepintf:
6893                                  ; end important
6894
6895                                  ; 17/10/2022
6896                                  ; 28/12/2018 - Retro DOS v4.0
6897
6898                                  ; (MSDOS 6.0, MSINIT.ASM, 1991)
6899                                  ;
6900                                  ; we will check if the system has ibm extended keyboard by
6901                                  ; looking at a byte at 40:96. if bit 4 is set, then extended keyboard
6902                                  ; is installed, and we are going to set keyrd_func to 10h, keysts_func to 11h
6903                                  ; for the extended keyboard function. use cx as the temporary register.
6904
6905                                          ; 21/12/2022
6906                                          ; ds = 0, cx = 0
6907                                          ;xor    cx, cx
6908                                          ;mov    ds, cx
6909
6910 00001E2E 8A0E9604                        mov     cl, [496h]      ; get keyboard flag
6911
6912                                          ; 21/12/2022
6913 00001E32 0E                              push    cs
6914 00001E33 1F                              pop     ds
6915
6916 00001E34 F6C110                          test    cl, 10h         ; extended keyboard ?
6917 00001E37 740A                            jz      short org_key   ; no, original keyboard
6918
6919                                          ; 21/12/2022
6920                                          ; ds = cs
6921 00001E39 C606[7E04]10                    mov     byte [keyrd_func], 10h ; extended keyboard
6922 00001E3E C606[7F04]11                    mov     byte [keysts_func], 11h
6923                                          ;mov    byte [cs:keyrd_func], 10h ; extended keyboard
6924                                          ;mov    byte [cs:keysts_func], 11h
6925                                                                  ; change for extended keyboard functions
6926                                  org_key:
6927
6928                                  ; 02/06/2018 - Retro DOS v3.0
6929
6930                                  ;*************************************************************
6931                                  ;   will initialize the number of drives
6932                                  ;   after the equipment call (int 11h) bits 6&7 will tell
6933                                  ;   the indications are as follows:
6934                                  ;
6935                                  ;   bits    7       6       drives
6936                                  ;           0       0       1
6937                                  ;           0       1       2
6938                                  ;           1       0       3
6939                                  ;           1       1       4
6940                                  ;*************************************************************
6941
6942                                          ; 21/12/2022
6943                                          ; ds = cs
6944                                          ;push   cs
6945                                          ;pop    ds
6946                                          ;push   cs
```

```
                                        ;pop    es
                                
00001E43 E8C70B                         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
00001E46 C706[0000][4B08]               mov     word [hdrv_pat], harddrv
                                
00001E4C 58                             pop     ax ; *          ; number of floppies and FAT ID
00001E4D 30E4                           xor     ah, ah          ; chuck fat id byte
00001E4F A2[7500]                       mov     [drvmax], al    ; remember which drive is hard disk
00001E52 A2[2501]                       mov     [dsktnum], al   ; and set initial number of drives
00001E55 D1E0                           shl     ax, 1
00001E57 0106[601A]                     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))
                                        ; ........
00001E5B 1E                             push    ds
00001E5C B800F0                         mov     ax, 0F000h
00001E5F 8ED8                           mov     ds, ax
                                
00001E61 813EEAFF434F                   cmp     word [0FFEAh], 'CO' ; 'COMPAQ'
00001E67 751F                           jne     short skip_mode2
00001E69 813EECFF4D50                   cmp     word [0FFECh], 'MP'
00001E6F 7517                           jne     short skip_mode2
00001E71 813EEEFF4151                   cmp     word [0FFEEh], 'AQ'
00001E77 750F                           jne     short skip_mode2
                                
00001E79 B800E4                         mov     ax, 0E400h      ; get advanced system info (COMPAQ ROMBIOS)
00001E7C CD15                           int     15h
00001E7E 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
00001E80 83CB40                         or      bx, 40h         ; enable mode 2 (dual harddisk controller)
00001E83 B880E4                         mov     ax, 0E480h       ; set advanced system info (COMPAQ ROMBIOS)
00001E86 CD15                           int     15h
                                skip_mode2:
00001E88 1F                             pop     ds
                                        ; ........
                                
00001E89 B280                           mov     dl, 80h
00001E8B B408                           mov     ah, 8
00001E8D 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
00001E8F 7204                           jc      short enddrv
00001E91 8816[5F1A]                     mov     [hnum], dl
                                enddrv:
                                        ; 21/12/2022
00001E95 0E                             push    cs
00001E96 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)
                                
00001E97 30D2                           xor     dl, dl
                                
                                        ; 21/12/2022
                                        ; ds = cs
                                        ;push   cs
                                        ;pop    ds
                                
00001E99 C606[2C01]09                   mov     byte [eot], 9
00001E9E 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.
                                
00001EA1 803E[131A]01                   cmp     byte [fakefloppydrv], 1
00001EA6 750B                           jnz     short loop_drive
00001EA8 8B3D                           mov     di, [di]        ; [di+BDS.link]
                                                                ; di <- first bds link
00001EAA 8B3D                           mov     di, [di]        ; [di+BDS.link]
                                                                ; di <- second bds link
00001EAC C705FFFF                       mov     word [di], 0FFFFh ; -1 ; set end of link
00001EB0 E98801                         jmp     dohard          ; allocate/initialise bds for harddrives
                                ;---------------------------------------------------------------------------
                                
                                loop_drive:
00001EB3 3A16[7500]                     cmp     dl, [drvmax]
00001EB7 7203                           jb      short got_more
00001EB9 E97B01                         jmp     done_drives
                                ;---------------------------------------------------------------------------
                                
                                got_more:
                                        ; 10/12/2023
                                        ;xor    cx, cx          ; zero all flags
00001EBC 8B3D                           mov     di, [di]        ; [di+BDS.link]
                                                                ; get next bds
                                        ; ........
                                        ; 10/12/2023 - Retro DOS v5.0
                                        ; (PCDOS 7.1 IBMBIO.COM BIOSDATA:2046h)
00001EBE 83FFFF                         cmp     di, 0FFFFh      ; end of link ?
00001EC1 7516                           jne     short not_last_bds
```

```
00001EC3 88D0                           mov     al, dl          ; drive number (0 based)
00001EC5 98                             cbw
00001EC6 01C0                           add     ax, ax
00001EC8 05[3C05]                       add     ax, dskdrvs
00001ECB A3[601A]                       mov     [last_dskdrv_table], ax
00001ECE 8B3E[621A]                     mov     di, [end_of_bdss]
00001ED2 E8020B                         call    xinstall_bds
00001ED5 FE0E[7500]                     dec     byte [drvmax]
                                not_last_bds:
                                        ; ........

00001ED9 B600                           mov     dh, 0           ; ff48tpi
                                                                ; set form factor to 48      tpi
00001EDB C606[101A]28                   mov     byte [num_cyln], 40 ; 40 tracks per side

                                        ; 21/12/2022
                                        ;push   ds
00001EE0 57                             push    di
00001EE1 52                             push    dx
                                        ;push   cx ; not necessary (10/12/2023)
00001EE2 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
00001EE3 52                             push    dx  ; dl = drive number

00001EE4 B408                           mov     ah, 8
00001EE6 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
00001EE8 58                             pop     ax  ; al = drive number
00001EE9 7303                           jnc     short chk_drv_type
00001EEB 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)
                                        ;
00001EEE 80FB10                         cmp     bl, 10h         ; ATAPI Removable Media Device
00001EF1 7554                           jne     short not_atapi_removable

                                        ; save ds:si
00001EF3 1E                             push    ds
                                        ;push   si      ; not necessary (10/12/2023)

00001EF4 88C2                           mov     dl, al
00001EF6 83EC1A                         sub     sp, 26
00001EF9 31C0                           xor     ax, ax ; 0
00001EFB 50                             push    ax
00001EFC B81E00                         mov     ax, 30
00001EFF 50                             push    ax
00001F00 89E6                           mov     si, sp          ; DS:SI = segment:offset pointer to Result Buffer
00001F02 16                             push    ss
00001F03 1F                             pop     ds
00001F04 B448                           mov     ah, 48h
00001F06 CD13                           int     13h             ; DISK - IBM/MS Extension
                                                                ; GET DRIVE PARAMETERS (DL - drive, DS:SI - buffer)
00001F08 7239                           jc      short ext_gdp_err
00001F0A 8B4408                         mov     ax, [si+8]      ; physical number of heads
00001F0D A3[0E1A]                       mov     [num_heads], ax
00001F10 8B4404                         mov     ax, [si+4]      ; physical number of cylinders
00001F13 A3[101A]                       mov     [num_cyln], ax
00001F16 8A440C                         mov     al, [si+0Ch]    ; physical number of sectors per track
00001F19 A2[121A]                       mov     [sec_trk], al
00001F1C 3A06[2C01]                     cmp     al, [eot]
00001F20 7603                           jbe     short _eotok
00001F22 A2[2C01]                       mov     [eot], al

                                _eotok:                 ; 10/12/2023
                                        ;xor    al, al
00001F25 31C9                           xor     cx, cx ; 0
00001F27 F6440210                       test    byte [si+2], 10h ; information flags
                                                                ; bit 4 = Device has change line support
00001F2B 7403                           jz      short not_chgline_sup
                                        ;or     al, 2           ; change line support
00001F2D 80C902                         or      cl, 2
                                not_chgline_sup:
00001F30 83C41E                         add     sp, 30
                                        ;pop    si      ; (10/12/2023)
00001F33 1F                             pop     ds
                                        ;
00001F34 07                             pop     es      ; es=cs=ds (21/12/2022)
                                        ;pop    cx      ; (10/12/2023)
00001F35 5A                             pop     dx
00001F36 5F                             pop     di
                                        ;pop    ds      ; (21/12/2022)

                                        ; 10/12/2023
00001F37 F6C102                         test    cl, 2
                                        ;test   al, 2
                                        ;jz     short gotother_j
00001F3A 7450                           jz      short gotother
                                        ;or     cl, al
00001F3C C606[7700]01                   mov     byte [fhave96], 1 ; Device has change line support
                                gotother_j:
00001F41 EB49                           jmp     short gotother
                                ext_gdp_err:
00001F43 83C41E                         add     sp, 30
                                        ;pop    si      ; (10/12/2023)
00001F46 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.

00001F47 80FD00                         cmp     ch, 0           ; if ch=0, then       cl,dh=0 too.
00001F4A 7505                           jnz     short pfr_ok

                                        ;mov    ch, 39          ; rom gave wrong info.
```

```
7195                                            ;mov    cl, 9           ; let's default to 360k.
7196                                            ; 21/12/2022
7197 00001F4C B90927                            mov     cx, 2709h
7198 00001F4F B601                              mov     dh, 1
7199                                    pfr_ok:
7200                                            ;inc    dh              ; make number of heads 1-based
7201                                            ;mov    [num_heads], dh       ; save parms returned by rom
7202                                            ; 10/12/2023
7203 00001F51 86F2                              xchg    dl, dh
7204 00001F53 30F6                              xor     dh, dh
7205 00001F55 42                                inc     dx              ; make number of heads 1-based
7206 00001F56 8916[0E1A]                        mov     [num_heads], dx
7207
7208                                            ;inc    ch              ; make number of cylinders 1-based
7209                                            ;and    cl, 3Fh
7210                                            ;mov    [sec_trk], cl
7211                                            ;mov    [num_cyln], ch ; assume less than 256 cylinders!!
7212                                            ; 10/12/2023
7213 00001F5A 88CA                              mov     dl, cl
7214 00001F5C 80E23F                            and     dl, 3Fh
7215 00001F5F 8816[121A]                        mov     [sec_trk], dl
7216 00001F63 86E9                              xchg    cl, ch
7217 00001F65 D0C5                              rol     ch, 1
7218 00001F67 D0C5                              rol     ch, 1
7219 00001F69 80E503                            and     ch, 3
7220 00001F6C 41                                inc     cx              ; make number of cylinders 1-based
7221 00001F6D 890E[101A]                        mov     [num_cyln], cx
7222
7223                                    ; make sure that eot contains the max number of sec/trk in system of floppies
7224
7225                                            ;mov    cl, [sec_trk] ; 10/12/2023
7226                                            ;cmp    cl, [eot]       ; may set carry
7227                                            ;;jbe   short eot_ok
7228                                            ;; 09/12/2022
7229                                            ;;jne   short eotok  ; wrong ! 14/08/2023
7230                                            ;; 14/08/2023
7231                                            ;jbe    short eotok
7232                                            ;mov    [eot], cl
7233                                            ; 10/12/2023
7234 00001F71 3A16[2C01]                        cmp     dl, [eot] ; dl = [sec_trk]
7235 00001F75 7604                              jbe     short eotok
7236 00001F77 8816[2C01]                        mov     [eot], dl
7237                                    ;eot_ok:
7238                                    eotok:
7239                                            ; 10/12/2023
7240                                            ; !!!
7241                                            ; (following pops are moved to 'chk_changeline' procedure)
7242                                            ;pop    es ; es=cs=ds ; 21/12/2022
7243                                            ;;pop   cx      ; (10/12/2023)
7244                                            ;pop    dx
7245                                            ;pop    di
7246
7247                                            ; 21/12/2022
7248                                            ;pop    ds
7249
7250                                    ; Check     for presence of       changeline
7251
7252                                    ; 10/12/2023
7253                                    %if 0
7254                                            ; 10/12/2023
7255                                            ;xor    cx, cx  ; 0
7256                                            ;push   cx
7257                                            push    dx
7258
7259                                            mov     ah, 15h
7260                                            int     13h             ; DISK - DISK -       GET TYPE (AT,XT2,XT286,CONV,PS)
7261                                                                    ; DL = drive ID
7262                                                                    ; Return: CF set on error, AH =      disk type (3 = hard drive)
7263                                                                    ; CX:DX = number of sectors on the media
7264                                            ; 10/12/2023
7265                                            pop     dx
7266                                            ;pop    cx
7267                                            mov     cx, 0 ; 12/12/2023
7268                                            jc      short changeline_done
7269                                            cmp     ah, 2           ; check for presence of       changeline
7270                                            jnz     short changeline_done
7271
7272                                    ; we have a drive with change line support.
7273
7274                                            or      cl, 2           ; fchangeline
7275                                                                    ; signal type
7276                                            mov     byte [fhave96], 1 ; remember that we have 96tpi disks
7277                                    %endif
7278                                            ; 10/12/2023
7279 00001F7B E83800                            call    chk_changeline
7280                                            ;jc     short changeline_done
7281
7282                                    ; we now try to set up the form factor for the types of media that we know
7283                                    ; and can recognise. for the rest, we set the form factor as "other".
7284
7285                                    changeline_done:
7286 00001F7E 803E[101A]28                      cmp     byte [num_cyln], 40
7287 00001F83 750B                              jnz     short try_80
7288 00001F85 803E[121A]09                      cmp     byte [sec_trk], 9
7289 00001F8A 765F                              jbe     short nextdrive
7290                                    gotother:
7291                                            ; 10/12/2023
7292                                            ; ch = 0, cl = 2 or 0
7293
7294 00001F8C B607                              mov     dh, 7           ; ffOther
7295                                                                    ; we have a "strange" medium
7296 00001F8E EB5B                              jmp     short nextdrive
7297                                    ;-----------------------------------------------------------------------------
7298
7299                                    ; 80 cylinders and 9 sectors/track => 720 kb device
7300                                    ; 80 cylinders and 15 sec/trk => 96 tpi medium
7301
7302                                    try_80:
7303 00001F90 803E[101A]50                      cmp     byte [num_cyln], 80
7304 00001F95 75F5                              jnz     short gotother
7305 00001F97 B609                              mov     dh, 9           ; ff288
7306                                                                    ; assume 2.88 MB drive
7307 00001F99 803E[121A]24                      cmp     byte [sec_trk], 36 ; is it   ?
7308 00001F9E 744B                              jz      short nextdrive       ; yeah, go update
7309
7310                                            ; 12/05/2019 (ff144 type will not be used -compatibility problem-)
7311                                            ; 08/01/2018 - Retro DOS v4.0 feature only ! for 1.44MB diskettes
7312                                            ;mov    dh, ff144
7313                                            ;cmp    byte [sec_trk], 18
7314                                            ;je     short nextdrive
7315
7316 00001FA0 803E[121A]0F                      cmp     byte [sec_trk], 15
7317 00001FA5 740B                              jz      short got96
7318
```

```
00001FA7 803E[121A]09                   cmp     byte [sec_trk], 9
00001FAC 75DE                           jnz     short gotother

00001FAE B602                           mov     dh, 2           ; ffSmall
00001FB0 EB39                           jmp     short nextdrive
                        ; ---------------------------------------------------------------------------

                        got96:
00001FB2 B601                           mov     dh, 1           ; ff96tpi
00001FB4 EB35                           jmp     short nextdrive

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

                                        ; 10/12/2023 - Retro DOS v5.0 IO.SYS/IBMBIO.COM
                                        ; check change line feature (and set fhave96 if there is)
                                        ; (common procedure for 'eotok:' and 'noparmsfromrom:')
                        chk_changeline:
00001FB6 59                             pop     cx ; near call return address

                                        ; (pop es, dx, di for 'eotok' and 'noparmsfromrom' procs)
00001FB7 07                             pop     es ; es=cs=ds ; 21/12/2022
                                        ;pop    cx      ; (10/12/2023)
00001FB8 5A                             pop     dx
00001FB9 5F                             pop     di ; BDS address/offset

00001FBA 51                             push    cx ; near call return address

                                        ;xor    cx, cx ; 0
                                        ;push   cx
00001FBB 52                             push    dx

00001FBC B415                           mov     ah, 15h
00001FBE CD13                           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
00001FC0 5A                             pop     dx
                                        ;pop    cx
00001FC1 B90000                         mov     cx, 0
00001FC4 720D                           jc      short chk_chgl_1

00001FC6 80FC02                         cmp     ah, 2           ; is there changeline?
00001FC9 7508                           jne     short chk_chgl_2 ; *

00001FCB 80C902                         or      cl, 2
                                        ;or     cl, ah ; 2
00001FCE C606[7700]01                   mov     byte [fhave96], 1 ; fchangeline
                                        ; cf = 0
                        chk_chgl_1:
                        chk_chgl_2:
00001FD3 C3                             retn

                        ;chk_chgl_2:            ; *
                        ;               ; 10/12/2023
                        ;               ; ah = 1 ; harddisk type (ah = 3) return not possible here for floppies
                        ;               ;stc
                        ;               ; cf = 1
                        ;               retn

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

                        ; we have an old rom, so we either have a 48tpi or 96tpi drive. if the drive
                        ; has changeline, we assume it is a 96tpi, otherwise we treat it as a 48tpi.

                        noparmsfromrom:
                                        ; 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
                        ; 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 nextdrive

                                        cmp     ah, 2           ; is there changeline?
                                        jnz     short nextdrive

                                        or      cl, 2
                                        mov     byte [fhave96], 1 ; fchangeline
                        %endif
                                        ; 10/12/2023
00001FD4 E8DFFF                         call    chk_changeline
00001FD7 7212                           jc      short nextdrive

                                        ; change line support, [fhave96] = 1

00001FD9 C606[101A]50                   mov     byte [num_cyln], 80
00001FDE B601                           mov     dh, 1           ; ff96tpi
00001FE0 B00F                           mov     al, 15
00001FE2 3A06[2C01]                     cmp     al, [eot]
00001FE6 7603                           jbe     short nextdrive
00001FE8 A2[2C01]                       mov     [eot], al
                        ; ---------------------------------------------------------------------------

                        nextdrive:
                                        ; 10/12/2023
                                        ; ch = 0, cl = 2 or 0

00001FEB 80C920                         or      cl, 20h         ; fi_own_physical
                                                                ; set this true       for all drives
00001FEE 88D7                           mov     bh, dl          ; save int13 drive number

                        ; we need to do special things if we have a single drive system and are setting
                        ; up a logical drive. it needs to have the same int13 drive number as its
```

```
                                ; 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.
00001FF0 803E[7800]02                   cmp     byte [single], 2
00001FF5 7505                           jnz     short not_special
00001FF7 FECF                           dec     bh              ; int13 drive number same for logical drive
00001FF9 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) ; *
00001FFC A1[0E1A]                       mov     ax, [num_heads]
                                        ;mov    [di+36h], ax    ; [di+BDS.rheads]
00001FFF 894552                         mov     [di+52h], ax    ; [di+BDS.rheads] ; *
00002002 31C0                           xor     ax, ax          ; *
00002004 A0[121A]                       mov     al, [sec_trk]
                                        ;mov    [di+34h], ax    ; [di+BDS.rsecpertrack]
00002007 894550                         mov     [di+50h], ax    ; [di+BDS.rsecpertrack] ; *
                                        ;mov    [di+23h], cx    ; [di+BDS.flags]
0000200A 894D3F                         mov     [di+3Fh], cx    ; [di+BDS.flags] ; *
                                        ;mov    [di+22h], dh    ; [di+BDS.formfactor]
0000200D 88753E                         mov     [di+3Eh], dh    ; [di+BDS.formfactor] ; *
00002010 885505                         mov     [di+5], dl      ; [di+BDS.drivelet]
00002013 887D04                         mov     [di+4], bh      ; [di+BDS.drivenum]
                                        ;mov    bl, [num_cyln]
                                        ;mov    [di+25h], bl    ; [di+BDS.cylinders]
                                        ; 10/12/2023
00002016 A1[101A]                       mov     ax, [num_cyln]
00002019 894541                         mov     [di+41h], ax    ; [di+BDS.cylinders] ; *

0000201C 803E[7800]01                   cmp     byte [single], 1 ; Special case for single drive system
00002021 7510                           jnz     short no_single
                                        ;mov    byte [single], 2 ; Don't forget we have
                                                                 ; single drive system
                                        ; 10/12/2023
00002023 FE06[7800]                     inc     byte [single]  ; [single] = 2
                                        ; 18/12/2022
00002027 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]
0000202A 094D3F                         or      [di+3Fh], cx    ; [di+BDS.flags] ; *
                                                                ; save flags
0000202D 8B3D                           mov     di, [di]        ; [di+BDS.link]
                                                                ; move to next BDS in list
0000202F FEC2                           inc     dl              ; add a number
00002031 EBB8                           jmp     short nextdrive       ; Use same info       for BDS as previous
                                ; ---------------------------------------------------------------------------

                                no_single:
                                        ;inc    dl
                                        ; 18/12/2022
00002033 42                             inc     dx
00002034 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
00002037 C705FFFF                       mov     word [di], -1  ; set link to null

                                ; set up all the hard drives in   the system

                                dohard:
0000203B 8A36[5F1A]                     mov     dh, [hnum]
0000203F 08F6                           or      dh, dh          ; done if no hardfiles
00002041 7459                           jz      short static_configure
00002043 B280                           mov     dl, 80h
                                dohard1:
00002045 52                             push    dx
00002046 8B3E[621A]                     mov     di, [end_of_bdss]
0000204A 8A1E[7500]                     mov     bl, [drvmax]
0000204E B700                           mov     bh, 0           ; first primary partition (or active)
00002050 E89902                         call    sethard
00002053 7208                           jc      short hardfile_err
00002055 E86A09                         call    dmax_check      ; error if already 26 drives
00002058 7303                           jnb     short hardfile_err
0000205A E87A09                         call    xinstall_bds    ; insert new bds into linked list
                                hardfile_err:
0000205D 5A                             pop     dx
                                        ;inc    dl              ; next hard drive
                                        ; 12/12/2023
0000205E 42                             inc     dx
0000205F FECE                           dec     dh
00002061 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).

00002063 E8CF07                         call    domini          ; for setting up mini disks, if found

                                ; -- begin added section

00002066 8A36[5F1A]                     mov     dh, [hnum]      ; we already know this is >0
0000206A B280                           mov     dl, 80h
                                dohardx1:
0000206C B701                           mov     bh, 1           ; do all subsequent primary partitions
                                dohardx2:
0000206E 52                             push    dx
0000206F 53                             push    bx
00002070 8B3E[621A]                     mov     di, [end_of_bdss]
00002074 8A1E[7500]                     mov     bl, [drvmax]
00002078 E87102                         call    sethard
0000207B 720E                           jc      short dohardx4 ; move to next hardfile       if error
0000207D E84209                         call    dmax_check      ; make sure <=26 drives
00002080 7309                           jnb     short dohardx4 ; skip if error
00002082 E85209                         call    xinstall_bds    ; insert new bds into linked list
00002085 5B                             pop     bx              ; get partition       number
00002086 5A                             pop     dx              ; restore physical drive counts
00002087 FEC7                           inc     bh
00002089 EBE3                           jmp     short dohardx2 ; keep looping until we       fail
                                ; ---------------------------------------------------------------------------

                                dohardx4:
0000208B 5B                             pop     bx              ; unjunk partition number from stack
```

```
7567 0000208C 5A                            pop     dx              ; restore physical drive counts
7568                                        ;inc    dl              ; next hard drive
7569                                        ; 12/12/2023
7570 0000208D 42                            inc     dx
7571 0000208E FECE                          dec     dh
7572 00002090 75DA                          jnz     short dohardx1
7573
7574                                ; -- end changed section
7575
7576                                ;--------------------------------------------------------------------------
7577                                ; if more than 2 diskette drives on the system, then it is necessary to remap
7578                                ; the bds chain to adjust the logical drive num (drive letter) with greater
7579                                ; than two diskette drives
7580                                ;
7581                                ; new scheme:      if more than 2 disktte drives, first map the bds structure
7582                                ;           as usual and then rescan the bds chain to adjust the  drive
7583                                ;           letters. to do this, scan for disk drives and assign logical
7584                                ;           drive number starting from 2 and then rescan diskette drives
7585                                ;           and assign next to the last logical drive number of last disk
7586                                ;           drive to the 3rd and 4th diskette drives.
7587                                ;--------------------------------------------------------------------------
7588
7589 00002092 803E[2501]02                    cmp     byte [dsktnum], 2 ; >2 diskette drives
7590                                        ;jbe    short static_configure ; no - no need for remapping
7591 00002097 7603                          jbe     short no_remap
7592 00002099 E8D401                        call    remap           ; remap bds chain to adjust driver letters
7593                                no_remap:
7594
7595                                ; End of drive initialization.
7596
7597                                ; ----------------------------------------------------------------------
7598
7599                                ;we now decide, based on the configurations available so far, what
7600                                ;code or data we need to keep as a stay resident code. the following table
7601                                ;shows the configurations under consideration. they are listed in the order
7602                                ;of their current position memory.
7603                                ;
7604                                ;configuration will be done in two ways:
7605                                ;
7606                                ;first, we are going to set "static configuration". static configuration will
7607                                ;consider from basic configuration to endof96tpi configuration. the result
7608                                ;of static configuration will be the address the dynamic configuration will
7609                                ;use to start with.
7610                                ;
7611                                ;secondly, "dynamic cofiguration" will be performed. dynamic configuration
7612                                ;involves possible relocation of code or data. dynamic configuration routine
7613                                ;will take care of bdsm tables and at rom fix module thru k09 suspend/resume
7614                                ;code individually. after these operation, [dosdatasg] will be set.
7615                                ;this will be the place sysinit routine will relocate msdos module for good.
7616
7617                                ; -- begin changed section
7618                                ;
7619                                ;   1.      basic configuration for msbio (endfloppy)
7620                                ;   2.   end96tpi  ; a system that supports "change line error"
7621                                ;   3.      end of bdss   ; end of bdss for hard disks
7622                                ;   4.      endatrom      ;some of at rom fix module.
7623                                ;   5.      endcmosclockset;supporting program for cmos clock write.
7624                                ;   6.      endk09        ;k09 cmos clock module to handle suspend/resume operation.
7625                                ;
7626
7627                                ; 02/10/2022 - Retro DOS v4.0 (MSDOS v5.0 IO.SYS)
7628
7629                                static_configure:
7630 0000209C 8B3E[621A]                     mov     di, [end_of_bdss]
7631 000020A0 81FF[4C08]                     cmp     di, bdss        ; 19/10/2022
7632                                        ;cmp    di, offset bdss         ; did we allocate any hard drive bdss?
7633 000020A4 750D                          jnz     short dynamic_configure         ; that's the end, then
7634                                        ; 18/10/2022
7635 000020A6 BF[4C08]                      mov     di, end96tpi
7636                                        ;mov    di, offset harddrv ; end96tpi
7637                                                                ; keep everything up to       end96tpi
7638 000020A9 803E[7700]00                  cmp     byte [fhave96], 0
7639 000020AE 7503                          jnz     short dynamic_configure
7640
7641 000020B0 BF[3808]                      mov     di, endfloppy
7642                                dynamic_configure:
7643                                        ; 20/12/2022
7644                                        ;push   cs
7645                                        ;pop    es
7646
7647                                        ; 10/12/2023
7648                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2268h)
7649                                        ; (MSDOS 6.22 IO.SYS - BIOSDATA:1C34h)
7650 000020B3 FC                            cld     ; clear direction flag is not necessary here !?
7651                                                ; because there will not be a running program
7652                                                ; which will set direction flag as backward (std)
7653
7654                                ; -- end changed section
7655
7656                                        ; 21/12/2022
7657                                        ; ds = cs <> es
7658                                        ; ss = 0
7659                                        ; sp = 700h
7660
7661                                        ; 13/12/2023
7662 000020B4 BE00F0                        mov     si, 0F000h
7663 000020B7 8EC6                          mov     es, si          ; ES -> ROM BIOS segment
7664
7665 000020B9 803E[AF05]FC                  cmp     byte [model_byte], 0FCh ; AT ?
7666                                        ;jnz    short checkcmosclock
7667                                        ; 10/12/2023
7668 000020BE 751E                          jnz     short checkcompaqbug ; no
7669 000020C0 803E[5F1A]00                  cmp     byte [hnum], 0 ; No hard file?
7670                                        ;jz     short checkcmosclock
7671 000020C5 7417                          jz      short checkcompaqbug
7672 000020C7 97                            xchg    ax, di          ; save allocation pointer in ax
7673                                        ; 13/12/2023
7674                                        ;mov    si, 0F000h
7675                                        ;mov    es, si          ; ES -> ROM BIOS segment
7676 000020C8 BE[681A]                      mov     si, bios_date   ; "01/10/84"
7677 000020CB BFF5FF                        mov     di, 0FFF5h      ; ROM BIOS string is at       F000:FFF5
7678 000020CE B90900                        mov     cx, 9           ; bdate_l
7679                                                                ; Only patch ROM for bios 01/10/84
7680 000020D1 F3A6                          repe cmpsb              ; check for date + zero       on end
7681 000020D3 97                            xchg    ax, di          ; restore allocation pointer
7682
7683                                ; M015 -- begin changes
7684
7685                                        ;jnz    short checkcmosclock
7686                                        ; 02/10/2022
7687 000020D4 7508                          jnz     short checkcompaqbug
7688
7689                                ; install at rom fix
7690
```

```
                                        ; 19/10/2022
                                        ;mov    cx, offset endatrom
000020D6 B9[2018]                       mov     cx, endatrom
                                        ;mov    si, offset ibm_disk_io
000020D9 BE[F216]                       mov     si, ibm_disk_io
000020DC 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
000020DE 26813EEAFF434F                 cmp     word [es:0FFEAh], 'CO'
                                        ;cmp    word ptr es:0FFEAh, 'OC' ; look for COMPAQ
000020E5 754B                           jnz     short not_compaq_patch
000020E7 26813EECFF4D50                 cmp     word [es:0FFECh], 'MP'
                                        ;cmp    word ptr es:0FFECh, 'PM'
000020EE 7542                           jnz     short not_compaq_patch
000020F0 26813EEEFF4151                 cmp     word [es:0FFEEh], 'AQ'
                                        ;cmp    word ptr es:0FFEEh, 'QA'
000020F7 7539                           jnz     short not_compaq_patch

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

000020F9 26A1FBFF                       mov     ax, [es:0FFFBh]       ; get year
000020FD 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)

000020FF 3D3638                         cmp     ax, 3836h ; 02/10/2022 (NASM syntax)
                                        ;cmp    ax, '86'        ; 3836h
                                                                ; is it 86?
00002102 772E                           ja      short not_compaq_patch
00002104 7218                           jb      short do_compaq_patch
00002106 26A1F5FF                       mov     ax, [es:0FFF5h]       ; get month
0000210A 86C4                           xchg    ah, al
0000210C 3D3830                         cmp     ax, 3038h ; 02/10/2022 (NASM syntax)
                                        ;cmp    ax, '08'        ; 3038h
                                                                ; is it 08?
0000210F 7721                           ja      short not_compaq_patch
00002111 720B                           jb      short do_compaq_patch
00002113 26A1F8FF                       mov     ax, [es:0FFF8h]       ; get day
00002117 86C4                           xchg    ah, al
00002119 3D3430                         cmp     ax, 3034h ; 02/10/2022 (NASM syntax)
                                        ;cmp    ax, '04'        ; 3034h
                                                                ; is it 04?
0000211C 7314                           jnb     short not_compaq_patch

                                do_compaq_patch:
0000211E B9[3D18]                       mov     cx, end_compaq_i13hook
                                        ;mov    si, endatrom
                                        ; 11/12/2023
00002121 BE[2018]                       mov     si, compaq_disk_io ; endatrom

                                install_int13_patch:
00002124 0E                             push    cs
00002125 07                             pop     es
                                        ; 18/10/2022
00002126 893E[B400]                     mov     [Orig13], di   ; set new rom bios int 13 vector
0000212A 8C0E[B600]                     mov     [Orig13+2], cs
0000212E 29F1                           sub     cx, si         ; size of rom fix module
00002130 F3A4                           rep movsb              ; relocate it

                                ; M065 -- end changes

                                ; ---------------------------------------------------------------------------
                                not_compaq_patch:               ; M065
                                        ; 17/10/2022
                                checkcmosclock:
                                        ; 18/10/2022
00002132 0E                             push    cs
00002133 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:
00002134 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)

                                        ; ........

00002135 B80041                         mov     ax, 4100h       ; wait for any external event (al=0)
00002138 B304                           mov     bl, 4           ; wait for 4 clock ticks
0000213A F9                             stc                     ; Assume we will fail
0000213B 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
                                        ; ........

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

00002140 C606[7900]01                   mov     byte [fhavek09], 1
                                                                ; remember we have a k09 type
00002145 1E                             push    ds
00002146 31C0                           xor     ax, ax
00002148 8ED8                           mov     ds, ax

0000214A 893EB001                       mov     [6Ch*4], di
                                        ;mov    ds:1B0h, di     ; [6Ch*4]
                                                                ; new int 6Ch handler
                                        ;mov    word ptr ds:1B2h, cs ; [6Ch*4+2]
0000214E 8C0EB201                       mov     word [6Ch*4+2], cs
00002152 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
00002153 BE[3E18]                       mov     si, int_6Ch
00002156 B9BE01                         mov     cx, endk09-int_6Ch ; 461 in PCDOS 7.1 IBMBIO.COM
00002159 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

0000215B 83C70F                         add     di, 15          ; round (up) to paragraph
                                        ; 10/12/2022
                                        ;shr    di, 1
                                        ;shr    di, 1
                                        ;shr    di, 1
                                        ;shr    di, 1
0000215E B104                           mov     cl, 4
00002160 D3EF                           shr     di, cl
                                        ; 10/12/2022
00002162 83C770                         add     di, 70h         ; KERNEL_SEGMENT (in fact: IO.SYS loading segment)
                                        ; 19/10/2022 - Temporary !
                                        ;db     81h, 0C7h, 70h, 0 ; add di, 0070h
00002165 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...)

00002169 A1[FC19]                       mov     ax, [drvfat]    ; get drive and       fat id
                                        ; 22/12/2022
                                        ; Note: SETDRIVES uses AL (drive number) only
0000216C 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
0000216F 0E                             push    cs              ; simulate far call
```

```
                                        call    call_bios_code ; get bds for drive
                                                ; 06/04/2024
                                                ; es:di = BDS address
00002173 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)
00002176 0E                             push    cs
00002177 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
0000217A 268B5D06                       mov     bx, [es:di+6]  ; [di+BDS.BDS_BPB.BPB_BYTESPERSECTOR]
0000217E 891E[0A1A]                     mov     [md_sectorsize], bx ; used by get_fat_sector proc.
                                        ; 11/12/2023 ; *
00002182 268A5D3B                       mov     bl, [es:di+3Bh]       ; [di+BDS.fatsiz] ; *
                                        ;mov    bl, [es:di+1Fh]       ; [di+BDS.fatsiz]
                                                                ; get size of fat on media
00002186 881E[081A]                     mov     [fbigfat], bl
0000218A 268A4D08                       mov     cl, [es:di+8]
0000218E 268B4517                       mov     ax, [es:di+17h] ; [di+BDS.BDS_BPB.BPB_HIDDENSECTORS]
00002192 2906[FE19]                     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)
00002196 268B4519                       mov     ax, [es:di+19h] ; [di+BDS.BDS_BPB.BPB_HIDDENSECTORS+2]
0000219A 1906[001A]                     sbb     [First_Data_Sector+2], ax ; *
                                        ;sbb    [bios_h], ax   ; subtract upper 16 bits of sector num
                                ; ------
0000219E 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
000021A0 1E                             push    ds
                                        ;xor    di, di
                                        ;mov    ds, di
000021A1 31DB                           xor     bx, bx ; 0
000021A3 8EDB                           mov     ds, bx
000021A5 8B1E3A05                       mov     bx, [53Ah]
                                        ;mov    bx, ds:53Ah           ; (First cluster of the 2nd dir entry
                                                                ; of root directory in the buffer at 500h)
000021A9 1F                             pop     ds
000021AA 8B36[801A]                     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
000021AE A0[081A]                       mov     al, [fbigfat]
000021B1 50                             push    ax              ; (*) save fbigfat flags
000021B2 A0[FC19]                       mov     al, [drvfat]
000021B5 0A06[821A]                     or      al, [Boot_Drv]
000021B9 757B                           jnz     short boot_drv_fixed ; hard disk
                                boot_drv_removable:                     ; calculate cluster count and set fbig or fbigbig flag
000021BB 53                             push    bx              ; for removable drives
```

```
8063 000021BC 51                                push    cx
8064                                            ; 28/12/2023
8065                                            ;push   dx ; (not necessary)
8066
8067                                            ; 12/12/2023
8068 000021BD 06                                push    es
8069 000021BE 1F                                pop     ds
8070
8071 000021BF 8B450E                            mov     ax, [di+0Eh]    ; [di+BDS.totalsecs16]
8072 000021C2 31D2                              xor     dx, dx
8073 000021C4 09C0                              or      ax, ax
8074 000021C6 7506                              jnz     short prep_totalsecs_ok
8075 000021C8 8B451B                            mov     ax, [di+1Bh]    ; [di+BDS.totalsecs32]
8076 000021CB 8B551D                            mov     dx, [di+1Dh]
8077                                    prep_totalsecs_ok:
8078 000021CE 2B4509                            sub     ax, [di+9]      ; [di+BDS.resectors]
8079 000021D1 83DA00                            sbb     dx, 0
8080 000021D4 50                                push    ax
8081 000021D5 52                                push    dx
8082 000021D6 8B5D11                            mov     bx, [di+11h]    ; [di+BDS.fatsecs16]
8083 000021D9 31C0                              xor     ax, ax
8084 000021DB 09DB                              or      bx, bx
8085 000021DD 7506                              jnz     short prep_fatsecs_ok
8086 000021DF 8B5D1F                            mov     bx, [di+1Fh]    ; [di+BDS.fatsecs32]
8087 000021E2 8B4521                            mov     ax, [di+21h]
8088                                    prep_fatsecs_ok:
8089 000021E5 8A4D0B                            mov     cl, [di+0Bh]    ; ax:bx = 32 bit count of FAT sectors
8090                                                                    ; [di+BDS.fats]
8091 000021E8 30ED                              xor     ch, ch
8092 000021EA F7E1                              mul     cx
8093 000021EC 91                                xchg    ax, cx
8094 000021ED F7E3                              mul     bx
8095 000021EF 01D1                              add     cx, dx
8096 000021F1 89C3                              mov     bx, ax          ; cx:bx = total (2*) fat sectors
8097 000021F3 5A                                pop     dx
8098 000021F4 58                                pop     ax              ; dx:ax = totals sectors - reserved sectors
8099 000021F5 29D8                              sub     ax, bx
8100 000021F7 19CA                              sbb     dx, cx          ; dx:ax = data sectors (includes root dir sectors)
8101 000021F9 8B5D0C                            mov     bx, [di+0Ch]    ; [di+BDS.direntries]
8102 000021FC 83C30F                            add     bx, 15          ; 16 directory entries per sector
8103                                                                    ; (round up sector count by adding 15)
8104 000021FF B104                              mov     cl, 4           ; (rounded) dir entries / 16
8105 00002201 D3EB                              shr     bx, cl
8106 00002203 31C9                              xor     cx, cx
8107 00002205 29D8                              sub     ax, bx
8108 00002207 19CA                              sbb     dx, cx          ; dx:ax = data sectors (except root directory sectors)
8109                                                                    ; (will be used for cluster count calculation)
8110 00002209 8A4D08                            mov     cl, [di+8]      ; [di+BDS.secperclus]
8111
8112                                            ; 12/12/2023
8113 0000220C 0E                                push    cs
8114 0000220D 1F                                pop     ds
8115
8116 0000220E 50                                push    ax              ; 32 bit division (data sectors / sector per cluster)
8117 0000220F 89D0                              mov     ax, dx
8118 00002211 31D2                              xor     dx, dx
8119 00002213 F7F1                              div     cx
8120 00002215 89C3                              mov     bx, ax
8121 00002217 58                                pop     ax
8122 00002218 F7F1                              div     cx
8123 0000221A 09DB                              or      bx, bx          ; 32 bit cluster count if bx > 0
8124 0000221C 7505                              jnz     short set_fbigbig_flag ; too big cluster number
8125 0000221E 83F8F6                            cmp     ax, 0FFF6h
8126 00002221 7207                              jb      short set_fbig_flag
8127                                    set_fbigbig_flag:
8128 00002223 800E[081A]20                      or      byte [fbigfat], 20h ; FAT32 ; fbigbig
8129 00002228 EB0A                              jmp     short set_fbig_flag_ok
8130                                    ; ---------------------------------------------------------------------------
8131
8132                                    set_fbig_flag:
8133 0000222A 3DF60F                            cmp     ax, 0FF6h       ; 4096-10
8134                                                                    ; is this 16-bit fat?
8135 0000222D 7205                              jb      short set_fbig_flag_ok ; no, small fat
8136 0000222F 800E[081A]40                      or      byte [fbigfat], 40h ; FAT16 ; fbig
8137                                    set_fbig_flag_ok:
8138                                            ; 28/12/2023
8139                                            ;pop    dx
8140 00002234 59                                pop     cx
8141 00002235 5B                                pop     bx
8142                                    boot_drv_fixed:
8143 00002236 31FF                              xor     di, di
8144
8145                                            ; cx = sectors/cluster
8146                                            ; si:bx = first cluster
8147                                            ; di = 0
8148
8149                                            ; ...........
8150                                    loadit:
8151 00002238 B80405                            mov     ax, SYSINITSEG ; 46Dh
8152                                            ;mov    ax, 544h        ; 11/12/2023 - PCDOS 7.1 IBMBIO.COM
8153                                            ;;mov   ax, 46Dh        ; sysinit segment
8154 0000223B 8EC0                              mov     es, ax
8155 0000223D 268E06[7302]                      mov     es, [es:CURRENTDOSLOCATION] ; 09/12/2022
8156                                            ;mov    es, [es:271h]
8157
8158 00002242 E86008                            call    getclus         ; read cluster at ES:DI (DI is updated)
8159
8160                                    ; ---------------------------------------------------------------------------
8161
8162                                            ; 13/12/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
8163                                            ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2431h)
8164
8165                                            ;test   byte [cs:fbigfat], 20h ; fbigbig ; FAT32 fs flag
8166 00002245 F606[081A]20                      test    byte [fbigfat], 20h ; fbigbig ; FAT32 fs flag
8167                                            ;jz     short iseof
8168                                            ; 06/04/2024
8169 0000224A 750D                              jnz     short eofbigbig
8170
8171                                    ; ---------------------------------------------------------------------------
8172                                    ; 06/04/2024
8173                                    %if 1
8174                                            ; 13/12/2023
8175                                    iseof:
8176                                            ;;test  byte [cs:fbigfat], fbig
8177                                            ;test   byte [cs:fbigfat], 40h ; fbig
8178                                            ; 12/12/2023
8179                                            ; ds = cs
8180 0000224C F606[081A]40                      test    byte [fbigfat], 40h ; fbig
8181 00002251 750C                              jnz     short eofbig
8182 00002253 81FBF70F                          cmp     bx, 0FF7h
8183 00002257 EB09                              jmp     short iseofx
8184                                    %endif
8185                                    ; ---------------------------------------------------------------------------
8186
```

```
                                eofbigbig:  ; si:bx = 32 bit cluster number
00002259 81FEFF0F                           cmp     si, 0FFFh
0000225D 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:
0000225F 83FBF7                             cmp     bx, 0FFF7h
                                iseofx:
00002262 72D4                               jb      short loadit   ; keep loading until cluster = eof
                                ; ---------------------------------------------------------------------------

                                            ; 06/04/2024
                                            ;call   setdrvparms    ;

                                            ; 28/12/2023
00002264 58                                 pop     ax             ; (*) restore fbigfat flags
                                                                   ; (after loading DOS kernel)
                                            ; 06/04/2024
                                            ;mov    [cs:fbigfat], al
00002265 A2[081A]                           mov     [fbigfat], al

00002268 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)

0000226B 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
00002270 8B3E[1901]                         mov     di, [start_bds]

                              ; search for 1st fixed disk physical drive num

                              drive_loop:
00002274 807D0480                           cmp     byte [di+4], 80h ; [di+BDS.drivenum]
                                                                     ; first hard disk??
00002278 7409                               jz      short fdrv_found ; yes,       continue
0000227A 8B3D                               mov     di, [di]       ; [di+BDS.link]
                                                                     ; get next bds,       assume segment
0000227C 83FFFF                             cmp     di, -1 ; 0FFFFh       ; last bds?
0000227F 75F3                               jnz     short drive_loop ; loop       if not
00002281 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:
00002283 B002                               mov     al, 2          ; start with logical drv num=2
                              fdrv_loop:
00002285 884505                             mov     [di+5], al     ; [di+BDS.drivelet]
00002288 8B3D                               mov     di, [di]       ; [di+BDS.link]
                                                                     ; ds:di--> next       bds
                                            ;inc    al             ; set num for next drive
```

```
                                        ; 18/12/2022
000022A A 40                            inc     ax
000022B B 83FFFF                        cmp     di, 0FFFFh      ; last hard drive ?
000022E E 75F5                          jnz     short fdrv_loop         ; no - assign more disk drives

                                ;---------------------------------------------------------------------------
                                ; now, rescan and find bds of 3rd floppy drive and assign next drive letter
                                ; in al to 3rd. if the current drive letter is past z, then do not allocate
                                ; any more.
                                ;---------------------------------------------------------------------------

                                        ;mov    di, [cs:start_bds] ; [start_bds]
                                        ; 15/12/2023
00002290 8B3E[1901]                     mov     di, [start_bds]         ; get first bds
00002294 8B3D                           mov     di, [di]        ; [di+BDS.link]
                                                                ; ds:di-->bds2
                                        ;mov    ah, [cs:dsktnum] ; get number of floppies to remap
00002296 8A26[2501]                     mov     ah, [dsktnum]
0000229A 80EC02                         sub     ah, 2           ; adjust for a:       & b:
                                remap_loop1:
0000229D 8B3D                           mov     di, [di]        ; [di+BDS.link]
                                                                ; set new num to next floppy
0000229F 884505                         mov     [di+5], al      ; [di+BDS.drivelet]
000022A2 FEC0                           inc     al              ; new number for next floppy
000022A4 FECC                           dec     ah              ; count down extra floppies
000022A6 75F5                           jnz     short remap_loop1

                                ; now we've got to adjust the boot drive if we reassigned it

                                        ; 15/12/2023
                                        ;mov    al, [cs:drvfat]
000022A8 A0[FC19]                       mov     al, [drvfat]
000022AB 3C02                           cmp     al, 2           ; is it a: or b: ?
000022AD 721D                           jb      short rmap_exit
                                        ;sub    al, [cs:dsktnum]
000022AF 2A06[2501]                     sub     al, [dsktnum]  ; is it one of the other floppies?
000022B3 7204                           jb      short remap_boot_flop ;       brif so

                                ; we've got to remap the boot hard drive
                                ; subtract the number of EXTRA floppies from it

000022B5 0402                           add     al, 2           ; bootdrv -= (dsktnum-2)
000022B7 EB04                           jmp     short remap_change_boot_drv
                                ; ---------------------------------------------------------------------------

                                ; we've got to remap the boot floppy.
                                ; add the number of hard drive partitions to it

                                remap_boot_flop:
                                        ;add    al, [cs:drvmax]        ; bootdrv += (drvmax-dsktnum)
                                        ; 15/12/2023
000022B9 0206[7500]                     add     al, [drvmax]
                                remap_change_boot_drv:
                                        ;mov    [cs:drvfat], al ; alter msdos.sys load drive
000022BD A2[FC19]                       mov     [drvfat], al
000022C0 FEC0                           inc     al
000022C2 1E                             push    ds
000022C3 BF0405                         mov     di, SYSINITSEG ; 46Dh
                                        ;mov    di, 544h        ; PCDOS 7.1 IBMBIO.COM
                                        ;;mov   di, 46Dh        ; SYSINIT segment
000022C6 8EDF                           mov     ds, di
000022C8 A2[9802]                       mov     [DEFAULTDRIVE], al
                                        ;mov    ds:296h, al     ; [SYSINIT+DEFAULT_DRIVE]
                                                                ; pass it to sysinit as       well
000022CB 1F                             pop     ds ; ds = cs
                                rmap_exit:
000022CC C3                             retn

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

                                ; 17/10/2022
                                ; 02/10/2022 - Retro DOS v4.0 (MSDOS 5.0 -actual-)
                                ; 28/12/2018 - Retro DOS v4.0 (MSDOS 6.21 -draft-)
                                ; 02/06/2018 - Retro DOS v3.0 (MSDOS 3.3)
                                ; 19/03/2018 - Retro DOS v2.0 (MSDOS 2.11)
                                ;************************************************
                                ; getboot - get the boot sector for a hard disk
                                ;
                                ; Reads the boot sector from a specified drive into
                                ; a buffer at the top of memory.
                                ;
                                ; dl = int13 drive number to read boot sector for
                                ;************************************************

                                ; 17/10/2022
                                bootbias equ 200h

                                getboot:        ; proc near

                                        ; 15/12/2023 - Retro DOS v5.0
                                        ;       (Modified PCDOS 7.1) IBMBIO.COM/IO.SYS
                                        ; ds = cs

                                        ; 08/04/2018
                                        ; Retro DOS v2.0 (IBMBIO.COM, IBMDOS 2.1)
                                        ; 28/03/2018 - MSDOS 6.0 - MSINIT.ASM, 1991
                                        ; 02/10/2022 - Retro DOS v4.0
                                        ;            (disassembled IO.SYS code of MSDOS 5.0)

                                        ;mov    ax, [cs:init_bootseg] ; 17/10/2022
                                        ; 15/12/2023
000022CD A1[061A]                       mov     ax, [init_bootseg]
000022D0 8EC0                           mov     es, ax

                                        ; 17/10/2022
000022D2 BB0002                         mov     bx, bootbias ; 200h
                                        ;mov    bx, 200h        ; bootbias
                                                                ; load BX, ES:BX is where sector goes
000022D5 B80102                         mov     ax, 201h
000022D8 30F6                           xor     dh, dh
000022DA B90100                         mov     cx, 1
000022DD 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
000022DF 7209                           jc      short erret
                                        ; 17/10/2022
000022E1 26813EFE0355AA                 cmp     word [es:bootbias+1FEh], 0AA55h
                                        ;cmp    word ptr es:3FEh, 0AA55h ; [es:bootbias+1FEh]
                                                                ; Dave Litton magic word?
000022E8 7401                           jz      short norm_ret ; yes
                                erret:
000022EA F9                             stc
                                norm_ret:
```

```
8435 000022EB C3                                retn
8436
8437                                    ; ============== S U B   R O U T I N E =======================================
8438
8439                                    ; 28/12/2018 - Retro DOS v4.0
8440
8441                                    ;******************************************************************************
8442                                    ;   sethard - generate bpb for a variable sized hard file. ibm has a
8443                                    ;   partitioned hard file; we must read physical sector 0 to determine where
8444                                    ;   our own logical sectors start. we also read in our boot sector to
8445                                    ;   determine version number
8446                                    ;
8447                                    ;   inputs: dl is rom drive number (80...)
8448                                    ;           bh is partition number (0....)
8449                                    ;           ds:di points to bds
8450                                    ;   outputs:       carry clear -> bpb is filled in
8451                                    ;           carry set   -> bpb is left uninitialized due to error
8452                                    ;   trashes (at least) si, cx
8453                                    ;   MUST PRESERVE ES:!!!!
8454                                    ;******************************************************************************
8455
8456                                            ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
8457                                            ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:24E9h)
8458
8459                                    sethard:    ; proc near
8460                                            ; 12/08/2023
8461                                            ; ds = cs = BIOSDATA
8462 000022EC 57                               push    di
8463 000022ED 53                               push    bx
8464                                            ;push   ds  ; ds = cs = BIOSDATA ; 12/08/2023
8465 000022EE 06                               push    es
8466 000022EF 885D05                           mov     [di+5], bl      ; [di+BDS.drivelet]
8467 000022F2 885504                           mov     [di+4], dl      ; [di+BDS.drivenum]
8468                                            ; 16/12/2023
8469 000022F5 804D3F01                         or      byte [di+3Fh], 1 ; PCDOS 7.1
8470                                            ;or     byte [di+23h], 1 ; [di+BDS.flags]
8471                                                                    ; fnon_removable
8472 000022F9 C6453E05                         mov     byte [di+3Eh], 5 ; PCDOS 7.1
8473                                            ;mov    byte [di+22h], 5 ; [di+BDS.formfactor]
8474                                                                    ; ffHardFile
8475 000022FD C606[081A]00                     mov     byte [fbigfat], 0 ; assume 12 bit FAT
8476 00002302 88FE                             mov     dh, bh          ; partition number
8477 00002304 52                               push    dx
8478 00002305 B408                             mov     ah, 8
8479 00002307 CD13                             int     13h             ; DISK - DISK -       GET CURRENT DRIVE PARAMETERS (XT,AT,XT286,CONV,PS)
8480                                                                    ; DL = drive number
8481                                                                    ; Return: CF set on error, AH =      status code, BL       = drive type
8482                                                                    ; DL = number of consecutive drives
8483                                                                    ; DH = maximum value for head number, ES:DI -> drive parameter
8484                                            ;inc    dh
8485                                            ; 16/12/2023 - Retro DOS v5.0
8486 00002309 88F2                             mov     dl, dh
8487 0000230B B600                             mov     dh, 0
8488 0000230D 42                               inc     dx
8489                                            ;mov    [di+15h], dh    ; [di+BDS.heads] ; get number of heads
8490 0000230E 895515                           mov     [di+15h], dx
8491 00002311 5A                               pop     dx
8492 00002312 7253                             jc      short setret    ; error if no hard disk
8493                                            ; 16/12/2023
8494                                            ;jc     short setret_j
8495
8496                                            ; 17/04/2024
8497                                            ;and    cx, 3Fh
8498                                            ;mov    [di+13h],cx
8499 00002314 80E13F                           and     cl, 3Fh
8500 00002317 884D13                           mov     [di+13h], cl    ; [di+BDS.secpertrack]
8501
8502 0000231A 52                               push    dx              ; save partition number
8503 0000231B E8AFFF                           call    getboot
8504 0000231E 5A                               pop     dx              ; restore partition number
8505 0000231F 7246                             jc      short setret
8506                                            ; 16/12/2023
8507                                            ;jnc    short chk_act_part
8508                                    ;setret_j:
8509                                            ;jmp    setret
8510
8511                                            ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
8512                                    chk_act_part:
8513 00002321 31DB                             xor     bx, bx ; 0
8514                                            ;;mov   [cs:ep_start_sector], bx
8515                                            ;;mov   [cs:ep_start_sector+2], bx
8516                                            ;mov    [cs:ep_hidden_secs], bx
8517                                            ;mov    [cs:ep_hidden_secs+2], bx
8518                                            ; 16/12/2023
8519                                            ; ds = cs
8520                                            ; 20/12/2023
8521                                            ;mov    [ep_start_sector], bx
8522                                            ;mov    [ep_start_sector+2], bx
8523 00002323 891E[8923]                       mov     [ep_hidden_secs], bx
8524 00002327 891E[8B23]                       mov     [ep_hidden_secs+2], bx
8525
8526 0000232B BBC203                           mov     bx, 3C2h        ; 1C2h+bootbias
8527
8528                                    ; The first 'active' partition is 00, the second is 01....
8529                                    ;   then the remainder of the 'primary' but non-active partitions
8530
8531                                    act_part:
8532 0000232E 26F647FC80                       test    byte [es:bx-4], 80h ; is the partition active?
8533 00002333 740B                             jz      short no_act    ; no
8534                                    ; 16/12/2023
8535                                    %if 0
8536                                            ; 16/12/2023
8537                                            ; reject if partitiontype != 1, 4, 6, 0Bh, 0Ch, 0Eh
8538                                            cmp     byte [es:bx], 1 ; FAT12
8539                                            jz      short got_good_act
8540                                            cmp     byte [es:bx], 4         ; FAT16 CHS (<= 32MB)
8541                                            jz      short got_good_act
8542
8543                                            ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
8544                                            cmp     byte [es:bx], 0Bh ; FAT32 CHS
8545                                            jz      short got_good_act
8546                                            cmp     byte [es:bx], 0Ch ; FAT32 LBA
8547                                            jz      short got_good_act
8548                                            cmp     byte [es:bx], 0Eh ; FAT16 LBA
8549                                            jz      short got_good_act
8550
8551                                            cmp     byte [es:bx], 6         ; FAT16 BIG CHS (> 32MB)
8552                                            jnz     short no_act
8553                                    ;%else
8554                                            ; 16/12/2023
8555                                            mov     al, [es:bx]     ; partition type
8556
8557                                            ; reject if partitiontype != 1, 4, 6, 0Bh, 0Ch, 0Eh
8558                                            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

00002335 E83300                         call    chk_partition_type
00002338 7506                           jne     short no_act

                                got_good_act:                           ; 11/08/2023
0000233A 08F6                           or      dh, dh          ; is this our target partition #?
                                                                ; (0 = first primary dos or active partition)
0000233C 744F                           jz      short set2      ; WE GOT THE ONE WANTED!!
0000233E FECE                           dec     dh              ; count down
                                no_act:
00002340 83C310                         add     bx, 16
00002343 81FB0204                       cmp     bx, 402h        ; 202h+bootbias
                                                                ; last entry done?
00002347 75E5                           jnz     short act_part ; no, process next entry

00002349 BBC203                         mov     bx, 3C2h        ; 1C2h+bootbias
                                                                ; restore original value of bx

                                ; Now scan the non-active partitions

                                get_primary:
0000234C 26F647FC80                     test    byte [es:bx-4], 80h
00002351 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

00002353 E81500                         call    chk_partition_type
00002356 7506                           jne     short not_prim

                                got_prim:
00002358 08F6                           or      dh, dh          ; is this our target partition?
0000235A 7431                           jz      short set2
0000235C FECE                           dec     dh
                                not_prim:
0000235E 83C310                         add     bx, 16
00002361 81FB0204                       cmp     bx, 402h        ; 202h+bootbias
00002365 75E5                           jne     short get_primary ; loop till we've gone through table
                                setret:
00002367 F9                             stc                     ; error return
00002368 E9C703                         jmp     ret_hard_err

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

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

                                chk_partition_type:
                                        ; 16/12/2023
0000236B 268A07                         mov     al, [es:bx]     ; partition type

                                        ; see if partitiontype == 1, 4, 6, 0Bh, 0Ch, 0Eh
0000236E 3C01                           cmp     al, 1           ; FAT12
00002370 7412                           je      short chk_ptype_retn
00002372 3C04                           cmp     al, 4           ; FAT16 CHS (<= 32MB)
00002374 740E                           je      short chk_ptype_retn

                                        ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
00002376 3C0B                           cmp     al, 0Bh         ; FAT32 CHS
00002378 740A                           je      short chk_ptype_retn
0000237A 3C0C                           cmp     al, 0Ch         ; FAT32 LBA
0000237C 7406                           je      short chk_ptype_retn
0000237E 3C0E                           cmp     al, 0Eh         ; FAT16 LBA
00002380 7402                           je      short chk_ptype_retn

00002382 3C06                           cmp     al, 6           ; FAT16 BIG CHS (> 32MB)
                                chk_ptype_retn:
                                        ; zf = 1 -> primary DOS partition
```

```
                                        ; zf = 0 -> not a primary DOS partition
00002384 C3                             retn
                                ; ----------------------------------------------------------------------------
                                        ; 16/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:25B6h)
                                ep_start_sector:
00002385 00000000                       dd 0
00002389 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)
0000238D 8816[091A]                     mov     [rom_drv_num], dl
                                        ;mov    [cs:rom_drv_num], dl
                                                ; save the rom bios drive number we are handling now.
00002391 268B4704                       mov     ax, [es:bx+4]  ; hidden sectors (start      sector)
00002395 268B5706                       mov     dx, [es:bx+6]
                                ; decrement the sector count by 1 to make it zero based. exactly 64k
                                ; sectors should be allowed
00002399 83E801                         sub     ax, 1
0000239C 83DA00                         sbb     dx, 0
0000239F 26034708                       add     ax, [es:bx+8]  ; sectors in partition
000023A3 2613570A                       adc     dx, [es:bx+10]
000023A7 7305                           jnc     short okdrive
000023A9 800E[081A]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
000023AE 0306[8923]                     add     ax, [ep_hidden_secs]
000023B2 1316[8B23]                     adc     dx, [ep_hidden_secs+2]
000023B6 7305                           jnc     short okdrive_1
000023B8 800E[081A]80                   or      byte [fbigfat], 80h ; ftoobig
                                okdrive_1:
000023BD 26803F0C                       cmp     byte [es:bx], 0Ch ; FAT32 LBA partition ID
000023C1 7418                           je      short set_lba_flag
000023C3 26803F0E                       cmp     byte [es:bx], 0Eh ; FAT16 LBA partition ID
000023C7 7412                           je      short set_lba_flag
000023C9 3B5513                         cmp     dx, [di+13h]   ; if dx > [di+BDS.secpertrack] then
000023CC 730D                           jnb     short set_lba_flag ; set LBA r/w flag
000023CE F77513                         div     word [di+13h]
000023D1 31D2                           xor     dx, dx
000023D3 F77515                         div     word [di+15h]
000023D6 3D0004                         cmp     ax, 400h        ; if ax (cylinder number) >= 1024
                                                                ;  set LBA r/w flag
000023D9 7204                           jb      short set3
                                set_lba_flag:
000023DB 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
                                        ;;;
000023DF 268B4704                       mov     ax, [es:bx+4]  ; start sector (LBA) of the partition
000023E3 268B5706                       mov     dx, [es:bx+6]
                                        ;add    ax, [cs:ep_hidden_secs]
                                        ;adc    dx, [cs:ep_hidden_secs+2]
                                        ; ds = cs
000023E7 0306[8923]                     add     ax, [ep_hidden_secs]
                                                                ; + hidden secs of the extd dos partion
000023EB 1316[8B23]                     adc     dx, [ep_hidden_secs+2]
000023EF 894517                         mov     [di+17h], ax   ; [di+BDS.hiddensecs]
000023F2 895519                         mov     [di+19h], dx   ; [di+BDS.hiddensecs+2]
000023F5 31C0                           xor     ax, ax ; 0
000023F7 89457B                         mov     [di+7Bh], ax   ; [di+BDS.bdsm_hidden_trks]
000023FA 89450E                         mov     [di+0Eh], ax   ; [di+BDS.totalsec16]
                                        ;;;
000023FD 268B570A                       mov     dx, [es:bx+10] ; # of sectors (high)
00002401 268B4708                       mov     ax, [es:bx+8]  ; # of sectors (low)
00002405 89551D                         mov     [di+1Dh], dx   ; [di+BDS.totalsecs32+2]
00002408 89451B                         mov     [di+1Bh], ax   ; [di+BDS.totalsecs32]
                                                                ; bpb->maxsec =       p->partitionlength
                                        ;cmp    dx, 0
                                        ;ja     short okdrive_1
                                        ; 16/12/2023
0000240B 09D2                           or      dx, dx
0000240D 7505                           jnz     short set3_read
0000240F 83F840                         cmp     ax, 64         ; if (p->partitionlength < 64)
                                        ;jb     short setret   ; return -1;
00002412 7264                           jb      short set3_err
                                ;okdrive_1:
                                        ; 16/12/2023
                                set3_read:
00002414 8B5519                         mov     dx, [di+19h]   ; [di+BDS.hiddensecs+2]
00002417 8B4517                         mov     ax, [di+17h]   ; [di+BDS.hiddensecs]
0000241A 31DB                           xor     bx, bx         ; boot sector number - for mini      disk
                                                                ; usually equal       to the # of sec/trk.
0000241C 8A5D13                         mov     bl, [di+13h]   ; [di+BDS.secpertrack]
0000241F 50                             push    ax
00002420 89D0                           mov     ax, dx
00002422 31D2                           xor     dx, dx
00002424 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
 00002426 A3[9E04]              mov     [saved_word], ax
 00002429 58                    pop     ax
 0000242A F7F3                  div     bx
 0000242C 88D1                  mov     cl, dl
 0000242E FEC1                  inc     cl
 00002430 8B5D15                mov     bx, [di+15h]   ; [di+BDS.heads]
 00002433 50                    push    ax
 00002434 31D2                  xor     dx, dx
                                ;mov    ax, [cs:saved_word]
 00002436 A1[9E04]              mov     ax, [saved_word]
 00002439 F7F3                  div     bx
                                ;mov    [cs:saved_word], ax
 0000243B A3[9E04]              mov     [saved_word], ax ; not necessary !? (ax must be 0)
 0000243E 58                    pop     ax
 0000243F F7F3                  div     bx              ; dl is head, ax is cylinder
                                ; 16/12/2023
 00002441 0E                    push    cs
 00002442 07                    pop     es ; (*)
 00002443 BB[5201]              mov     bx, disksector ; (**)
                                ;
 00002446 F6454004              test    byte [di+40h], 4 ; fLBArw ; LBA read/write flag
 0000244A 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
 0000244C BE[591B]              mov     si, dap_buffer ; empty_dap_buff
 0000244F 56                    push    si
 00002450 87FE                  xchg    si, di
                                ; si = BDS
                                ; di = DAP buffer
```

```
8931 00002452 B81000                        mov     ax, 16
8932 00002455 AB                            stosw           ; DAP size
8933 00002456 B001                          mov     al, 1
8934 00002458 AB                            stosw           ; sector (read) count
8935                                        ; buffer address
8936 00002459 89D8                          mov     ax, bx  ; offset disksector
8937 0000245B AB                            stosw
8938 0000245C 8CC0                          mov     ax, es  ; es=ds=cs = BIOSDATA segment
8939 0000245E AB                            stosw
8940                                        ; sector address (bits 0 to 31)
8941 0000245F 8B4417                        mov     ax, [si+17h] ; [di+BDS.hiddensectors]
8942 00002462 AB                            stosw
8943 00002463 8B4419                        mov     ax, [si+19h] ; [di+BDS.hiddensectors+2]
8944 00002466 AB                            stosw
8945                                        ; sector address bits 32 to 63 (0)
8946 00002467 31C0                          xor     ax, ax ; 0
8947 00002469 AB                            stosw
8948 0000246A AB                            stosw
8949                                        ;xchg   di, si
8950 0000246B 89F7                          mov     di, si
8951                                        ; di = BDS
8952 0000246D 5E                            pop     si ; DAP buffer address
8953
8954 0000246E 8A16[091A]                    mov     dl, [rom_drv_num] ; ds = cs
8955 00002472 B442                          mov     ah, 42h
8956 00002474 CD13                          int     13h             ; DISK - IBM/MS Extension
8957                                                                ; EXTENDED READ (DL - drive, DS:SI - disk address packet)
8958                                        ;pop    si ; *
8959 00002476 7324                          jnc     short set3_read_ok
8960                                set3_err:
8961                                        ;jmp    setret
8962 00002478 E9B702                        jmp     ret_hard_err
8963                                %endif
8964
8965                                set3_chs_read:
8966 0000247B 837D7901                      cmp     word [di+79h], 1 ; [di+BDS.bdsm_ismini] ; check for mini disk
8967 0000247F 7503                          jnz     short oknotmini
8968 00002481 03457B                        add     ax, [di+7Bh]    ; [di+BDS.bdsm_hidden_trks]
8969                                        ;;;
8970
8971                                oknotmini:
8972                                ;*** end of added logic for mini disk
8973
8974 00002484 D0CC                          ror     ah, 1           ; move high two bits of cyl to high
8975 00002486 D0CC                          ror     ah, 1           ; two bits of upper byte
8976 00002488 80E4C0                        and     ah, 0C0h        ; turn off remainder of bits
8977 0000248B 08E1                          or      cl, ah          ; move two bits to correct spot
8978 0000248D 88C5                          mov     ch, al          ; ch iscylinder (low 8 bits)
8979                                                                ; cl is sector + 2 high bits of cylinder
8980 0000248F 88D6                          mov     dh, dl          ; dh is head
8981
8982                                        ; 12/08/2023 (ds=cs)
8983 00002491 8A16[091A]                    mov     dl, [rom_drv_num]
8984                                        ;mov    dl, [cs:rom_drv_num] ; dl is drive number
8985
8986                                ; cl is sector + 2 high bits of cylinder
8987                                ; ch is low 8 bits of cylinder
8988                                ; dh is head
8989                                ; dl is drive
8990
8991                                ; for convenience, we are going to read the logical boot sector
8992                                ; into cs:disksector area.
8993
8994                                ; read in boot sector using bios disk interrupt. the buffer where it
8995                                ; is to be read in is cs:disksector.
8996
8997                                        ; 16/12/2023
8998                                        ; es=ds=cs = BIOSDATA segment
8999                                        ; bx = disksector ; (**)
9000
9001                                        ;push   cs
9002                                        ;pop    es ; (*)
9003
9004                                        ;mov    bx, disksector ; for convenience,
9005                                                               ; we are going to read the logical boot sector
9006                                                               ; into cs:disksector area.
9007 00002495 B80102                        mov     ax, 201h
9008 00002498 CD13                          int     13h             ; DISK - READ SECTORS INTO MEMORY
9009                                                                ; AL = number of sectors to read, CH = track, CL = sector
9010                                                                ; DH = head, DL       = drive, ES:BX -> buffer to fill
9011                                                                ; Return: CF set on error, AH =      status, AL = number of sectors read
9012                                        ; 16/12/2023
9013 0000249A 72DC                          jc      short set3_err
9014
9015                                ; cs:disksec contains the boot sector. in theory, (ha ha) the bpb in this thing
9016                                ; is correct. we can, therefore, suck out all the relevant statistics on the
9017                                ; media if we recognize the version number.
9018
9019                                set3_read_ok:
9020                                        ; 11/08/2023
9021                                        ;mov    bx, disksector ; BIOSDATA:014Eh ; MSDOS 6.21 ; 11/08/2023
9022                                                               ; BIOSDATA:0152h ; PCDOS 7.1 IBMBIO.COM
9023                                        ; 18/12/2023
9024                                        ;push   bx ; +
9025                                        ;push   ax ; (not necessary)
9026
9027                                        ; 16/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
9028                                        ;;;
9029 0000249C 81BFFE0155AA                  cmp     word [bx+1FEh], 0AA55h
9030 000024A2 7541                          jne     short invalid_boot_record
9031                                        ; 16/12/2023
9032                                        ; 12/08/2023
9033                                        ; ds = cs = BIOSDATA segment ('disksector:' is in BIOSDATA)
9034 000024A4 803FE9                        cmp     byte [bx], 0E9h         ; is it a near jump?
9035 000024A7 740B                          je      short check_1_ok ; yes
9036 000024A9 803FEB                        cmp     byte [bx], 0EBh         ; is it a short jump?
9037 000024AC 7537                          jne     short invalid_boot_record ; no
9038 000024AE 807F0290                      cmp     byte [bx+2], 90h ; yes, is the next one a nop?
9039 000024B2 7531                          jne     short invalid_boot_record ; no, invalid bs ; 11/08/2023
9040                                check_1_ok:
9041 000024B4 837F1600                      cmp      word [bx+16h], 0 ; [bx+BPB_FATSz16]
9042                                        ;jz     short check_1  ; 16 bit FAT size is 0 if it is FAT32 bs
9043                                        ; 16/12/2023
9044 000024B8 740E                          jz      short check_2  ; FAT32 bs
9045
9046                                        ; FAT16 or FAT12 bs
9047
9048                                        ;push   ds
9049                                        ;push   si  ; (not necessary)
9050 000024BA 57                            push    di
9051                                        ; es=ds=cs = BIOSDATA segment
9052                                        ;push   es
9053                                        ;pop    ds
9054
```

```
9055                                            ;mov    cx, 28
9056 000024BB B90E00                            mov     cx, 14 ; *
9057 000024BE 8D7724                            lea     si, [bx+24h]   ; move offset 36 to 63
9058                                                                   ;       to offset 64 (28 bytes)
9059 000024C1 8D7F40                            lea     di, [bx+40h]   ; boot sector offset 64
9060 000024C4 FC                                cld     ; (not necessary, 'std' is not used before here)
9061                                            ;rep movsb
9062 000024C5 F3A5                              rep movsw ; *
9063 000024C7 5F                                pop     di
9064                                            ;pop    si
9065                                            ;pop    ds
9066                                            ;;;
9067                                    ; 16/12/2023
9068                                    %if 0
9069                                    ;check_1:
9070                                            ; 12/08/2023
9071                                            ; ds = cs = BIOSDATA segment ('disksector:' is in BIOSDATA)
9072                                            cmp     byte [bx], 0E9h
9073                                            ;cmp    byte [cs:bx], 0E9h ; is it a near jump?
9074                                            je      short check_1_ok ; yes
9075                                            cmp     byte [bx], 0EBh
9076                                            ;cmp    byte [cs:bx], 0EBh ; is it a short jump?
9077                                            jne     short invalid_boot_record ; no
9078                                            cmp     byte [bx+2], 90h
9079                                            ;cmp    byte [cs:bx+2], 90h ; yes, is the next one a nop?
9080                                            jne     short invalid_boot_record ; no, invalid bs ; 11/08/2023
9081                                    check_1_ok:
9082                                    %endif
9083
9084                                    ; 18/12/2023
9085                                    %if 0
9086                                            ; 14/08/2023
9087                                    check_2:
9088                                            mov     bx, disksector+11 ; disksector+EXT_BOOT.BPB
9089                                            ;mov    bx, 159h        ; disksector+EXT_BOOT.BPB
9090                                                                    ; point to the bpb in the boot record
9091                                            ;mov    al, [cs:bx+10] ; [bx+EBPB.MEDIADESCRIPTOR]
9092                                            mov     al, [bx+10] ; 12/08/2023
9093                                                                    ; get the mediadescriptor byte
9094                                            and     al, 0F0h        ; mask off low nibble
9095                                            cmp     al, 0F0h        ; is high nibble = 0Fh?
9096                                            jne     short invalid_boot_record ; no, invalid boot record
9097                                            ;cmp    word [cs:bx], 512 ; [bx+EBPB.BYTESPERSECTOR]
9098                                            cmp     word [bx], 512 ; 12/08/2023
9099                                            jne     short invalid_boot_record ; invalidate non 512 byte sectors
9100
9101                                    check2_ok:                      ; yes, mediadescriptor ok.
9102                                            mov     al, [bx+2] ; 12/08/2023
9103                                            ;mov    al, [cs:bx+2]  ; now make sure that
9104                                                                    ; the sectorspercluster is
9105                                                                    ; a power of 2
9106                                                                    ;
9107                                                                    ; [bx+EBPB.SECTORSPERCLUSTER]
9108                                                                    ; get the sectorspercluster
9109                                    %endif
9110                                            ;;;
9111                                    check_2:
9112                                            ; 18/12/2023
9113                                            ; bx = disksector
9114 000024C8 8A4715                            mov     al, [bx+21]    ; [bx+EXT_BOOT.BPB+EBPB.MEDIADESCRIPTOR]
9115                                                                    ; get the mediadescriptor byte
9116 000024CB 24F0                              and     al, 0F0h        ; mask off low nibble
9117 000024CD 3CF0                              cmp     al, 0F0h        ; is high nibble = 0Fh?
9118 000024CF 7514                              jne     short invalid_boot_record ; no, invalid boot record
9119 000024D1 817F0B0002                        cmp     word [bx+11], 512 ; [bx+EXT_BOOT.BPB+EBPB.BYTESPERSECTOR]
9120 000024D6 750D                              jne     short invalid_boot_record ; invalidate non 512 byte sectors
9121
9122                                    check2_ok:  ; yes, mediadescriptor ok.
9123 000024D8 8A470D                            mov     al, [bx+13]    ; now make sure that
9124                                                                    ; the sectorspercluster is
9125                                                                    ; a power of 2
9126                                                                    ;
9127                                                                    ; [bx++EXT_BOOT.BPB+EBPB.SECTORSPERCLUSTER]
9128                                                                    ; get the sectorspercluster
9129                                            ;;;
9130
9131 000024DB 08C0                              or      al, al         ; is it zero?
9132 000024DD 7406                              jz      short invalid_boot_record ; yes, invalid boot record
9133
9134                                    ck_power_of_two:
9135 000024DF D0E8                              shr     al, 1          ; shift until first bit emerges
9136 000024E1 73FC                              jnc     short ck_power_of_two
9137 000024E3 7406                              jz      short valid_boot_record
9138
9139                                    invalid_boot_record:
9140                                            ; 18/12/2023
9141                                            ;pop    ax
9142                                            ;pop    bx ; +
9143 000024E5 E96001                            jmp     unknown        ; jump to invalid boot record
9144                                                                    ; unformatted or illegal media.
9145                                    ; 16/12/2023
9146                                    ; ---------------------------------------------------------------------------
9147                                    ;   ; 12/08/2023
9148                                    ;setret_brdg:
9149                                    ;           jmp     setret
9150                                    ; ---------------------------------------------------------------------------
9151
9152                                    unknown3_0_j:
9153 000024E8 E96101                            jmp     unknown3_0     ; legally formatted media,
9154                                                                    ; although, content might be bad.
9155                                    ; ---------------------------------------------------------------------------
9156
9157                                    valid_boot_record:
9158                                            ; 18/12/2023
9159                                            ;pop    ax
9160                                            ;pop    bx ; +
9161
9162                                            ; 18/12/2023
9163                                            ; bx = offset disksector ; +
9164
9165                                    ; Signature found. Now check version.
9166
9167                                            ; 14/08/2023
9168 000024EB 817F08322E                        cmp     word [bx+8], '2.'
9169                                            ;cmp    word [cs:bx+8], '2.' ; 03/10/2022 (NASM syntax)
9170                                            ;;cmp   word ptr cs:[bx+8], 2E32h ; '2.'
9171 000024F0 7506                              jne     short try5
9172 000024F2 807F0A30                          cmp     byte [bx+10], '0'
9173                                            ;cmp    byte [cs:bx+0Ah], '0' ; 03/10/2022 (NASM syntax)
9174                                            ;;cmp   byte ptr cs:[bx+0Ah], 30h ; '0'
9175                                            ; 12/08/2023
9176                                            ;jnz    short try5
9177                                            ;jmp    short copybpb
9178 000024F6 7425                              je      short copybpb
```

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

                                try5:
000024F8 E83B02                             call    cover_fdisk_bug

                                ; see if it is an os2 signature

                                            ; 12/08/2023
                                            ; ds = cs = BIOSDATA segment
000024FB 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.'
00002500 750C                               jne     short no_os2
00002502 8A4707                             mov     al, [bx+7] ; 12/08/2023
                                            ;mov    al, [cs:bx+7]  ; 17/10/2022 (NASM syntax)
00002505 2C31                               sub     al, '1'
                                            ;sub    al, 31h        ; '1'
00002507 24FE                               and     al, 0FEh
00002509 7412                               jz      short copybpb  ; accept either       '1' or '2'
0000250B E93A01                             jmp     unknown
                                ; ---------------------------------------------------------------------------

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

                                no_os2:
                                            ; 12/08/2023
                                            ; ds = cs = BIOSDATA
0000250E 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.'
00002513 72D3                               jb      short unknown3_0_j ; must be 2.1 boot record.
                                                                       ; do not trust it, but still legal.
00002515 7506                               jnz     short copybpb  ; honor os2 boot record
                                                                   ; or dos 4.0 version
00002517 807F0A31                           cmp     byte [bx+10], '1' ; 12/08/2023
                                            ;cmp    byte [cs:bx+10], '1'
                                            ;;cmp   byte ptr cs:[bx+0Ah], 31h ; '1'
0000251B 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
0000251D BE[5D01]                           mov     si, disksector+11
                                            ;sub    ch, ch ; ; (ch may be > 0)
00002520 29C9                               sub     cx, cx ; 0
                                            ;mov    cl, [disksector+16] ; BPB_NumFATs
00002522 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)
00002525 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:
0000252A 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]
0000252C 20C9                           and     cl, cl ; 0 ?
0000252E 7534                           jnz     short copybpb_fat ; a fat system

                                ; non fat based    media.

00002530 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
00002531 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
00002534 394C08                         cmp     [si+8], cx ; 0        ; [si+EBPB.TOTALSECTORS]
00002537 751C                           jnz     short already_nonz
                                                                ; how about big_total?
00002539 394C15                         cmp     [si+15h], cx ; 0      ; [si+EBPB.BIGTOTALSECTORS]
0000253C 7517                           jnz     short already_nonz ; we're okay if any are != 0
0000253E 394C17                         cmp     [si+17h], cx ; 0      ; [si+EBPB.BIGTOTALSECTORS+2]
00002541 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.

00002543 8B4508                         mov     ax, [di+8]     ; [di+BDS.totalsecs16]
                                        ; 12/08/2023
                                        ;mov    [cs:si+8], ax  ; [cs:si+EBPB.TOTALSECTORS]
00002546 894408                         mov     [si+8], ax
00002549 8B4515                         mov     ax, [di+15h]   ; [di+BDS.totalsecs32]
                                        ;mov    [cs:si+15h], ax       ; [cs:si+EBPB.BIGTOTALSECTORS]
0000254C 894415                         mov     [si+15h], ax
0000254F 8B4517                         mov     ax, [di+17h]   ; [di+BDS.totalsecs32+2]
                                        ;mov    [cs:si+17h], ax       ; [cs:si+EBPB.BIGTOTALSECTORS+2]
00002552 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
00002555 B135                           mov     cl, 53  ; PCDOS 7.1 IBMBIO.COM
                                                        ; BDS.BPB size (25 + 28 for FAT32 parms)
00002557 F3A4                           rep movsb
                                        ;pop    ds
                                        ; 12/08/2023
                                        ; ds = cs
                                        ;pop    bp  ; ds (on top of stack) = BIOSDATA
00002559 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
0000255A 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.
0000255D 0E                             push    cs              ; simulate far call
0000255E E812F5                         call    call_bios_code
                                        ; 12/08/2023
                                        ; ds = cs = es
                                        ;push   es
                                        ;pop    ds
                                        ;pop    es
```

```
9427 00002561 E9C701                            jmp     goodret
9428
9429                                    ; ----------------------------------------------------------------------------
9430
9431                                    ; ****** cas ---
9432                                    ; IBM DOS 3.30 doesn't seem to mind that the TOTAL_SECTORS and
9433                                    ; BIG_TOTAL_SECTORS field in the boot sector are 0000. This
9434                                    ; happens with some frequency -- perhaps through some OS/2 setup
9435                                    ; program. We haven't actually been COPYING the TOTAL_SECTORS
9436                                    ; from the boot sector into the DPB anyway, we've just been using
9437                                    ; it for calculating the fat size. Pretty scary, huh? For now,
9438                                    ; we'll go ahead and copy it into the DPB, except in the case
9439                                    ; that it equals zero, in which case we just use the values in
9440                                    ; the DPB from the partition table.
9441
9442                                    ; 17/10/2022
9443                                    ;MOVMEDIAIDS equ mov_media_ids - DOSBIOSEG_2C7h ; (751h for MSDOS 5.0 IO.SYS)
9444                                    ;CLEARIDS equ clear_ids - DOSBIOSEG_2C7h ; (5D9h for MSDOS 5.0 IO.SYS)
9445                                    ; 09/12/2022
9446                                    MOVMEDIAIDS equ mov_media_ids
9447                                    CLEARIDS equ clear_ids
9448                                    ; 11/09/2023
9449                                    CLEARIDS_X equ clear_ids_x
9450
9451                                    copybpb_fat:
9452                                                ; 17/12/2023
9453                                                ; ch = 0, cl = number of FATs
9454                                                ; 10/12/2022
9455                                                ; (number of FATs optimization)
9456                                                ; SI = disksector+11
9457                                                ; 17/10/2022
9458                                                ;mov    si, disksector+11
9459                                                ;;mov   si, 159h       ; disksector+EXT_BOOT.BPB
9460                                                                      ; cs:si -> bpb in boot
9461                                                ; 17/12/2023
9462                                                ; dx = 0
9463                                                ; 18/04/2024 (BugFix)
9464 00002564 31D2                                  xor     dx, dx
9465
9466                                                ; 12/08/2023
9467                                                ; ds = cs = BIOSDATA segment (0070h)
9468 00002566 8B4408                               mov     ax, [si+8]
9469                                                ;mov    ax, [cs:si+8]  ; [cs:si+EBPB.TOTALSECTORS]
9470                                                                       ; get totsec from boot sec
9471 00002569 09C0                                 or      ax, ax
9472 0000256B 7514                                 jnz     short copy_totsec ; if non zero, use that
9473 0000256D 8B4415                               mov     ax, [si+15h] ; 12/08/2023
9474                                                ;mov    ax, [cs:si+15h]       ; [cs:si+EBPB.BIGTOTALSECTORS]
9475                                                                       ; get the big version
9476                                                                       ; (32 bit total       sectors)
9477 00002570 8B5417                               mov     dx, [si+17h] ; 12/08/2023
9478                                                ;mov    dx, [cs:si+17h]       ; [cs:si+EBPB.BIGTOTALSECTORS+2]
9479                                                ; 10/12/2022
9480                                                ; (number of FATs optimization)
9481                                                ; CL = number of FATs (2 or 1)
9482 00002573 89D3                                 mov     bx, dx         ; see if it is a big zero
9483 00002575 09C3                                 or      bx, ax
9484 00002577 7508                                 jnz     short copy_totsec
9485                                                ; screw it. it was bogus.
9486 00002579 8B451B                               mov     ax, [di+1Bh]   ; [di+BDS.totalsecs32]
9487 0000257C 8B551D                               mov     dx, [di+1Dh]   ; [di+BDS.totalsecs32+2]
9488 0000257F EB06                                 jmp     short fat_big_small
9489
9490                                                ;mov    cx, dx
9491                                                ;or     cx, ax         ; see if it is a big zero
9492                                                ;jz     short totsec_already_set ; screw it. it     was bogus.
9493                                    copy_totsec:
9494 00002581 89451B                               mov     [di+1Bh], ax   ; [di+BDS.totalsecs32]
9495                                                                       ; make DPB match boot sec
9496 00002584 89551D                               mov     [di+1Dh], dx   ; [di+BDS.totalsecs32+2]
9497
9498                                                ; 10/12/2022
9499                                    ;totsec_already_set:
9500                                                ;mov    ax, [di+1Bh]   ; [di+BDS.totalsecs32]
9501                                                ;mov    dx, [di+1Dh]   ; [di+BDS.totalsecs32+2]
9502
9503                                    ; determine fat entry size.
9504
9505                                    fat_big_small:
9506
9507                                    ;at this moment dx;ax = total sector number
9508
9509                                    ;Do not assume 1 reserved sector. Update the reserved sector field in BDS
9510                                    ;from the BPB on the disk
9511
9512                                                ; 12/08/2023
9513                                                ; ds = cs = BIOSDATA segment (0070h)
9514
9515 00002587 8B5C03                               mov     bx, [si+3]
9516                                                ;mov    bx, [cs:si+3]  ; [cs:si+EBPB.RESERVEDSECTORS]
9517                                                                       ; get #reserved_sectors       from BPB
9518 0000258A 895D09                               mov     [di+9], bx     ; [di+BDS.resectors]
9519                                                                       ; update BDS field
9520 0000258D 29D8                                 sub     ax, bx
9521 0000258F 83DA00                               sbb     dx, 0          ; update the count
9522                                                ; 12/08/2023
9523 00002592 8B5C0B                               mov     bx, [si+0Bh]
9524                                                ;mov    bx, [cs:si+0Bh]       ; [cs:si+EBPB.SECTORSPERFAT]
9525                                                                       ; bx = sectors/fat
9526 00002595 895D11                               mov     [di+11h], bx   ; [di+BDS.fatsecs]
9527                                                                       ; set in bds bpb
9528                                                ; 17/12/2023 - Retro DOS v5.0
9529                                                ;            (PCDOS 7.1 IBMBIO.COM)
9530 00002598 53                                   push    bx ; FAT sectors
9531 00002599 09DB                                 or      bx, bx
9532 0000259B 753A                                 jnz     short fat_16bit
9533
9534                                    ; 17/12/2023
9535                                    %if 0
9536                                                sub     ax, [si+19h]   ; FAT32 file system (PCDOS 7.1 BUG!)
9537                                                                       ; BPB.FATSz32
9538                                                sbb     dx, [si+1Bh]   ; BPB.FATSz32+2 (PCDOS 7.1 BUG!)
9539                                                ; dx:ax = partition size - (one FAT sectors + reserved secta)
9540                                                mov     bx, [si+19h]   ; BPB.FATSz32
9541                                                mov     [di+1Fh], bx   ; [di+BDS.fatsecs32]
9542                                                mov     bx, [si+1Bh]   ; BPB.FATSz32+2
9543                                                mov     [di+21h], bx   ; [di+BDS.fatsecs32+2]
9544                                                mov     bx, [si+1Dh]   ; BPB.BPB_ExtFlags
9545                                                mov     [di+23h], bx   ; [di+BDS.extflags]
9546                                                mov     bx, [si+1Fh]   ; BPB.FSVer
9547                                                mov     [di+25h], bx   ; [di+BDS.fsver]
9548                                                mov     bx, [si+21h]   ; BPB.RootClus
9549                                                mov     [di+27h], bx   ; [di+BDS.rootdirclust]
9550                                                mov     bx, [si+23h]   ; BPB.RootClus+2
```

```
                                        mov     [di+29h], bx   ; [di+BDS.rootdirclust+2]
                                        mov     bx, [si+25h]   ; BPB.FSInfo
                                        mov     [di+2Bh], bx   ; [di+BDS.fsinfo]
                                        mov     bx, [si+27h]   ; BPB.FSInfo+2
                                        mov     [di+2Dh], bx   ; [di+BDS.fsinfo+2]
                                        jmp     short fat_32bit       ; PCDOS 7.1 BUG! Erdogan Tan - 8/8/2023
                                                                      ; correct code (would be):
                                                                      ;   mov cl, [cs:si+05h] ; BPB_NumFATs
                                                                      ; sub_fat32_size:
                                                                      ;   sub ax, [cs:si+19h] ; BPB_FATSz32
                                                                      ;   sbb dx, [cs:si+1Bh] ; BPB_FATSz32+2
                                                                      ;   dec cl
                                                                      ;   jg short sub_fat32_size
                                                                      ;   jmp short fat_32bit
                                %endif
                                        ; 17/12/2023
                                        ; cl = BPB_NumFATs (2 or 1)
                                        ; ch = 0
000025D7 8B5C19                         mov     bx, [si+19h]   ; BPB.FATSz32
                                sub_fat32_size:
000025DA 29D8                           sub     ax, bx
000025DC 1B541B                         sbb     dx, [si+1Bh]   ; BPB.FATSz32+2
                                        ;dec    cl
000025DF 49                             dec     cx
000025E0 7FF8                           jg      short sub_fat32_size

000025E2 895D1F                         mov     [di+1Fh], bx   ; [di+BDS.fatsecs32]
000025E5 8B5C1B                         mov     bx, [si+1Bh]   ; BPB.FATSz32+2
000025E8 895D21                         mov     [di+21h], bx   ; [di+BDS.fatsecs32+2]

000025EB 8B5C1D                         mov     bx, [si+1Dh]   ; BPB.BPB_ExtFlags
000025EE 895D23                         mov     [di+23h], bx   ; [di+BDS.extflags]
000025F1 8B5C1F                         mov     bx, [si+1Fh]   ; BPB.FSVer
000025F4 895D25                         mov     [di+25h], bx   ; [di+BDS.fsver]
000025F7 8B5C21                         mov     bx, [si+21h]   ; BPB.RootClus
000025FA 895D27                         mov     [di+27h], bx   ; [di+BDS.rootdirclust]
000025FD 8B5C23                         mov     bx, [si+23h]   ; BPB.RootClus+2
00002600 895D29                         mov     [di+29h], bx   ; [di+BDS.rootdirclust+2]
00002603 8B5C25                         mov     bx, [si+25h]   ; BPB.FSInfo
00002606 895D2B                         mov     [di+2Bh], bx   ; [di+BDS.fsinfo]
00002609 8B5C27                         mov     bx, [si+27h]   ; BPB.FSInfo+2
0000260C 895D2D                         mov     [di+2Dh], bx   ; [di+BDS.fsinfo+2]
0000260F EB08                           jmp     short fat_32bit

                                fat_16bit:
                                        ; 17/12/2023 - Retro DOS v5.0
                                        ;              (PCDOS 7.1 IBMBIO.COM)
                                        ; 10/12/2022
                                        ; (number of FATs optimization)
                                        ; CL = number of FATs (2 or 1)
                                        ; CH = 0 ; 17/12/2023
                                        ;dec    cl ; *
                                        ; 18/12/2022
00002611 49                             dec     cx ; *
00002612 D3E3                           shl     bx, cl
                                        ;shl    bx, 1   ; =*?=  ; always 2 fats

00002614 29D8                           sub     ax, bx         ; sub # fat sectors
00002616 83DA00                         sbb     dx, 0
                                fat_32bit:
                                        ; 17/12/2023
00002619 8B5C06                         mov     bx, [si+6] ; 12/08/2023
                                        ;mov    bx, [cs:si+6]  ; [cs:si+EBPB.ROOTENTRIES]
                                                               ; # root entries
0000261C 895D0C                         mov     [di+0Ch], bx   ; [di+BDS.direntries]
                                                               ; set in bds bpb
0000261F B104                           mov     cl, 4
00002621 D3EB                           shr     bx, cl         ; div by 16 ents/sector
00002623 29D8                           sub     ax, bx         ; sub # dir sectors
00002625 83DA00                         sbb     dx, 0          ;
                                                               ; dx:ax now contains the
                                                               ; # of data sectors
                                        ; 17/12/2023
                                        ; ch = 0
                                        ;xor    cx, cx ; *
00002628 8A4C02                         mov     cl, [si+2] ; 12/08/2023
                                        ;mov    cl, [cs:si+2]  ; [cs:si+EBPB.SECTORSPERCLUSTER]
                                                               ; sectors per cluster
0000262B 884D08                         mov     [di+8], cl     ; [di+BDS.secperclus]
                                                               ; set in bios bpb
0000262E 50                             push    ax
0000262F 89D0                           mov     ax, dx
00002631 31D2                           xor     dx, dx
00002633 F7F1                           div     cx             ; cx = sectors per cluster
                                        ; 12/08/2023 (ds=cs)
                                        ;mov    [temp_h], ax
                                        ;;mov   [cs:temp_h], ax       ; [temp_h]:ax now contains the
                                                               ; # clusters.
                                        ; 17/12/2023
00002635 A3[9E04]                       mov     [saved_word], ax ; hw of cluster number
00002638 58                             pop     ax
00002639 F7F1                           div     cx
                                        ; 17/12/2023
                                        ;;cmp   word [cs:temp_h], 0
                                        ;cmp    word [temp_h], 0  ; 12/08/2023
                                        ;cmp    word [saved_word], 0 ; (*)
                                        ;ja     short toobig_ret ; too big cluster number

                                        ; 17/12/2023
                                        ;;;
0000263B 5B                             pop     bx ; FAT sectors (16 bit)
                                        ;and    bx, bx ; 0 ?
0000263C 09DB                           or      bx, bx ; 0 ?
0000263E 751F                           jnz     short chk_clnum_hw
                                                        ; 16 bit fat sectors > 0 ; FAT12 or FAT16 fs

00002640 813E[9E04]FF0F                 cmp     word [saved_word], 0FFFh
00002646 7503                           jne     short fat32_clust_limit
00002648 83F8F6                         cmp     ax, 0FFF6h     ; FAT32 cluster number limit: 0FFFFFF6h
                                fat32_clust_limit:
0000264B 772D                           ja      short short toobig_ret ; too big cluster number
0000264D 391E[9E04]                     cmp     [saved_word], bx ; 0 ?
                                        ;jnz    short fat16_clust_limit
00002651 7505                           jnz     short set_bigbig_flag ; 17/12/2023
                                fat16_clust_limit: ; 17/12/2023
00002653 83F8F6                         cmp     ax, 0FFF6h     ; FAT16 cluster number limit: 0FFF6h
                                ;fat16_clust_limit:
00002656 760E                           jna     short fat12_clust_limit ; jbe
                                set_bigbig_flag:   ; 17/12/2023
00002658 800E[081A]20                   or      byte [fbigfat], 20h ; fbigbig ; FAT32 fs
0000265D EB11                           jmp     short copymediaid
                                chk_clnum_hw:
0000265F 833E[9E04]00                   cmp     word [saved_word], 0 ; (*)
00002664 7714                           ja      short toobig_ret ; too big cluster number
```

```
                                        ;;;
                                fat12_clust_limit:
0000262C 3DF60F                         cmp     ax, 0FF6h       ; 4096-10
                                                                ; is this 16-bit fat?
0000262F 7205                           jb      short copymediaid ; no,       small fat
                                        ; 17/10/2022
00002631 800E[081A]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
00002636 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
00002639 0E                             push    cs              ; simulate far call
0000263A E836F4                         call    call_bios_code

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

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

                                toobig_ret:
                                        ; 12/08/2023 (ds=cs=BIOSDATA)
00002640 800E[081A]80                   or      byte [fbigfat], 80h ; ftoobig
                                        ;or     byte [cs:fbigfat], 80h ; ftoobig
                                                                ; too big (32 bit clust #) for FAT16
00002645 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)
00002648 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:
0000264C 8B551D                         mov     dx, [di+1Dh]    ; skip setting unformatted_media bit
                                                                ; [di+BDS.totalsecs32+2]
0000264F 8B451B                         mov     ax, [di+1Bh]    ; [di+BDS.totalsecs32]
00002652 BE[181A]                       mov     si, disktable2
                                scan:                                   ; 08/08/2023
                                        ;cmp    dx, [cs:si]     ; total sectors hw
                                        ; 12/08/2023 (ds=cs)
00002655 3B14                           cmp     dx, [si]
00002657 720C                           jb      short gotparm
00002659 7705                           ja      short scan_next
                                        ;cmp    ax, [cs:si+2]   ; total sectors lw
0000265B 3B4402                         cmp     ax, [si+2]
0000265E 7605                           jbe     short gotparm
                                scan_next:
00002660 83C60A                         add     si, 10          ; 5*2
00002663 EBF0                           jmp     short scan      ; covers upto 512 mb media
                                ; ---------------------------------------------------------------------------

                                gotparm:
00002665 8A4C08                         mov     cl, [si+8]      ; fat size for fbigfat flag
                                        ;or     ds:fbigfat, cl
                                        ; 17/10/2022
00002668 080E[081A]                     or      [fbigfat], cl   ; (fbig flag, 40h or 0) ; 08/08/2023
                                        ; 12/08/2023
                                        ; ds = cs = BIOSDATA
0000266C 8B4C04                         mov     cx, [si+4]
                                        ;mov    cx, [cs:si+4]   ; ch = number of sectors per cluster
                                                                ; cl = log base 2 of ch
0000266F 8B5406                         mov     dx, [si+6]
                                        ;mov    dx, [cs:si+6]   ; dx = number of root dir entries

                                ; now calculate size of fat table

00002672 89550C                         mov     [di+0Ch], dx    ; [di+BDS.direntries]
                                                                ; save number of (root)       dir entries
00002675 8B551D                         mov     dx, [di+1Dh]    ; [di+BDS.totalsecs32+2]
00002678 8B451B                         mov     ax, [di+1Bh]    ; [di+BDS.totalsecs32]
0000267B 886D08                         mov     [di+8], ch      ; [di+BDS.secperclus]
                                                                ; save sectors per cluster

                                        ; 17/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
0000267E F606[081A]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)
00002683 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 !!

00002685 31DB                           xor     bx, bx          ; 12 bit fat (FAT12 fs)
00002687 88EB                           mov     bl, ch
00002689 4B                             dec     bx
0000268A 01C3                           add     bx, ax          ; dx=0
0000268C D3EB                           shr     bx, cl          ; bx = 1+(bpb->maxsec+BDS.secperclus-1)/
0000268E 43                             inc     bx              ; BDS.secperclus
```

```
9799 0000268F 80E3FE                    and     bl, 0FEh        ; bx &= ~1; (=number of clusters)
9800 00002692 89DE                      mov     si, bx
9801 00002694 D1EB                      shr     bx, 1
9802 00002696 01F3                      add     bx, si          ; number of FAT bytes ; 08/08/2023
9803 00002698 81C3FF01                  add     bx, 511         ; bx += 511 + bx/2
9804 0000269C D0EF                      shr     bh, 1           ; bh >>= 1; (=bx/512)
9805 0000269E 887D11                    mov     [di+11h], bh    ; [di+BDS.fatsecs]
9806                                                            ; save number of fat sectors
9807 000026A1 EB6A                      jmp     short massage_bpb
9808                                ; ---------------------------------------------------------------------------
9809
9810                                ; for bigfat we do need to extend this logic to 32 bit sector calculation.
9811
9812                                dobig:
9813 000026A3 B104                      mov     cl, 4           ; 16 (2^4) directory entries per sector
9814 000026A5 52                        push    dx              ; save total sectors (high)
9815 000026A6 8B550C                    mov     dx, [di+0Ch]    ; [di+BDS.direntries]
9816 000026A9 D3EA                      shr     dx, cl          ; root dir sectors = BDS.direntries / 16;
9817 000026AB 29D0                      sub     ax, dx
9818 000026AD 5A                        pop     dx
9819 000026AE 83DA00                    sbb     dx, 0           ; dx:ax = total sectors - root dir sectors
9820 000026B1 83E801                    sub     ax, 1
9821 000026B4 83DA00                    sbb     dx, 0           ; dx:ax = t - r - d
9822                                                            ; total secs - reserved       secs - root dir       secs
9823 000026B7 B302                      mov     bl, 2
9824 000026B9 8A7D08                    mov     bh, [di+8]      ; [di+BDS.secperclus]
9825                                                            ; bx = 256 * BDS.secperclus + 2
9826
9827                                ; I don't understand why to add bx here!!!
9828
9829                                        ; 29/12/2018 - Erdogan Tan (Retro DOS v4.0)
9830                                        ; 27/09/2022
9831                                        ; (Microsoft FAT32 File      System Specification,
9832                                        ; December 2000, Page 21)
9833                                        ; TmpVal1 = DskSize - (BPB_ResvdSecCnt+RootrDirSectors)
9834                                        ; TmpVal2 = (256*BPB_SecPerClus)+BPB_NumFATs
9835                                        ; 8/8/2023 (Retro DOS v5.0)
9836                                        ; If(FATType == FAT32)
9837                                        ;   TmpVal2 = TmpVal2 / 2;
9838                                        ; FATsz = (TmpVal1+(TmpVal2-1))/TmpVal2
9839                                        ; 8/8/2023 (Retro DOS v5.0)
9840                                        ; If(FATType == FAT32) {
9841                                        ;   BPB_FATSz16 = 0;
9842                                        ;   BPB_FATSz32 = FATSz;
9843                                        ;} else {
9844                                        ;   BPB_FATSz16 = LOWORD(FATSz);
9845                                        ;/* there is no BPB_FATSz32 in a FAT16 BPB */
9846                                        ;}
9847                                                            ; dx:ax = TmpVal1, bx = TmpVal2
9848 000026BC 01D8                      add     ax, bx          ;
9849 000026BE 83D200                    adc     dx, 0           ; dx:ax = TmpVal1+TmpVal2
9850 000026C1 83E801                    sub     ax, 1
9851 000026C4 83DA00                    sbb     dx, 0           ; dx:ax = TmpVal1+TmpVal2-1
9852
9853                                        ;;;
9854                                        ; 17/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
9855 000026C7 F606[081A]20                  test    byte [fbigfat], 20h ; fbigbig (FAT32) flag
9856 000026CC 740D                          jz      short dobig1
9857
9858 000026CE D1EB                      shr     bx, 1           ; TmpVal2 = TmpVal2 / 2
9859                                                            ; dx:ax = TmpVal1+(2*TmpVal2)-1
9860 000026D0 83E81F                    sub     ax, 31          ; reserved sectors = 32 (for FAT32 fs) /// 1+31 = 32
9861 000026D3 83DA00                    sbb     dx, 0
9862 000026D6 29D8                      sub     ax, bx
9863 000026D8 83DA00                    sbb     dx, 0           ; dx:ax = TmpVal1+(2*TmpVal2)-TmpVal2-1
9864                                                            ;       = TmpVal1+(TmpVal2-1)
9865                                dobig1:
9866 000026DB 50                        push    ax              ; save lw of dividend
9867 000026DC 89D0                      mov     ax, dx          ; divide hw of dx:ax at first (as 1st stage)
9868 000026DE 31D2                      xor     dx, dx
9869 000026E0 F7F3                      div     bx              ; 32 bit division, dx:ax/bx
9870                                                            ; remainder in dx is hw of 2nd stage dividend
9871 000026E2 89C5                      mov     bp, ax          ; hw of quotient
9872 000026E4 58                        pop     ax              ; restore lw of dividend (of 1st stage)
9873                                        ;;;
9874
9875                                ; assuming dx in the table will never be bigger than bx.
9876
9877 000026E5 F7F3                      div     bx              ; BDS.fatsecs =
9878                                                            ; ceil((total-dir-res)/(256*BDS.secperclus+2))
9879 000026E7 894511                    mov     [di+11h], ax    ; [di+BDS.fatsecs]
9880                                                            ; number of fat       sectors
9881                                        ;;;
9882
9883                                        ; 18/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
9884 000026EA 8A1E[081A]                    mov     bl, [fbigfat]
9885 000026EE 885D3B                       mov     [di+3Bh], bl    ; [di+BDS.fatsiz] ; fat size flag
9886
9887 000026F1 F6C320                        test    bl, 20h         ; fbigbig (FAT32) flag
9888 000026F4 7410                          jz      short dobig2    ; not FAT32
9889
9890 000026F6 89451F                        mov     [di+1Fh], ax    ; [di+BDS.fatsecs32]
9891 000026F9 896D21                        mov     [di+21h], bp    ; [di+BDS.fatsecs32+2]
9892 000026FC C745110000                    mov     word [di+11h], 0 ; [di+BDS.fatsecs] = 0
9893                                                                ; clear 16 bit FAT size field
9894 00002701 C745092000                    mov     word [di+9], 32       ; [di+BDS.resectors]
9895                                                                ; set reserved sectors to 32 (FAT32 de facto)
9896                                dobig2:
9897                                        ;;;
9898
9899                                ; now, set the default filesys_id, volume label, serial number
9900
9901                                        ; 05/08/2023
9902                                        ; [di+1Fh] = [fbigfat]
9903                                        ;
9904                                        ;;mov   bl, ds:fbigfat
9905                                        ;; 17/10/2022
9906                                        ;mov    bl, [fbigfat]
9907                                        ;mov    [di+1Fh], bl    ; [di+BDS.fatsiz] ; fat      size flag
9908
9909                                        ; 12/08/2023
9910                                        ;push   ds ; ds = cs = BIOSDATA
9911
9912                                        ; 17/12/2023
9913                                        ; es = ds = cs
9914                                        ;push   ds
9915                                        ;pop    es
9916
9917                                        ; 12/08/2023
9918                                        ; ds = cs = BIOSDATA
9919                                        ;push   cs
9920                                        ;pop    ds
9921
9922                                        ; 18/12/2023 - Retro DOS v5.0
```

```
                                        ; bl = [fbigfat] (clear_ids_x uses bl value here)
                                        ; 11/09/2023
                                        ;mov    al, [fbigfat]
00002706 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
00002709 0E                             push    cs
0000270A 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
                                        ;
0000270D 8B551D                         mov     dx, [di+1Dh]    ; [di+BDS.totalsecs32+2]
00002710 8B451B                         mov     ax, [di+1Bh]    ; [di+BDS.totalsecs32]
                                        ; 11/09/2023
00002713 09D2                           or      dx, dx
00002715 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
00002717 395519                         cmp     [di+19h], dx ; 0
                                        ;ja     short goodret2
0000271A 770F                           ja      short goodret  ; 11/09/2023
0000271C 034517                         add     ax, [di+17h]   ; [di+BDS.hiddensecs]
                                        ;jb     short goodret
                                        ; 12/12/2022
                                        ;jc     short goodret
0000271F 7209                           jc      short goodret_clc ; 11/09/2023
00002721 8B451B                         mov     ax, [di+1Bh]   ; [di+BDS.totalsecs32]
00002724 89450E                         mov     [di+0Eh], ax   ; [di+BDS.totalsecs16]
                                        ;mov    word [di+1Bh], 0 ; [di+BDS.totalsecs32]
                                        ; 12/12/2022
00002727 89551B                         mov     [di+1Bh], dx ; 0
                                goodret_clc:
                                        ; 11/09/2023
0000272A F8                             clc
                                goodret:
                                        ;mov    bl, ds:fbigfat
                                        ; 11/09/2023
                                        ; 12/12/2022
                                        ; 17/10/2022
0000272B 8A1E[081A]                     mov     bl, [fbigfat]
                                        ; 17/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
0000272F 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:
00002732 07                             pop     es
                                        ;pop    ds      ; ds = cs = BIOSDATA ; 14/08/2023
00002733 5B                             pop     bx
00002734 5F                             pop     di
00002735 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
00002736 837F1600                       cmp     word [bx+16h],0
0000273A 742C                           jz      short cfb_retit ; FAT32 boot sector

                                        ; 18/12/2023
0000273C 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
00002740 7426                           je      short cfb_retit       ; if extended bpb, then       >= pc dos 4.00

00002742 817F073130                     cmp     word [bx+7], 3031h
                                        ;cmp    word [cs:bx+7], 3031h ; '10' ; os2 1.0 = ibm 10.0
00002747 7506                           jne     short cfb_chk_totalsecs ; 11/08/2023
```

```
10047 00002749 807F0A30                         cmp     byte [bx+10], '0'
10048                                           ;cmp    byte [cs:bx+10], '0'
10049 0000274D 7519                             jne     short cfb_retit
10050
10051                                   cfb_chk_totalsecs:
10052                                           ; 11/08/2023
10053                                   ; 18/12/2023
10054                                   %if 0
10055                                           ; 17/10/2022
10056                                           mov     si, disksector+11 ; 14Eh+0Bh
10057                                           ;mov    si, 159h        ; disksector+EXT_BOOT.BPB
10058                                           ; 12/08/2023
10059                                           cmp     word [si+8], 0
10060                                           ;cmp    word [cs:si+8], 0 ; [cs:si+EBPB.TOTALSECTORS]
10061                                                                   ; just to make sure.
10062                                           jz      short cfb_retit
10063                                           ;mov    ax, [cs:si+8]  ; [cs:si+EBPB.TOTALSECTORS]
10064                                           ;add    ax, [cs:si+11h]        ; [cs:si+EBPB.HIDDENSECTORS]
10065                                           ; 12/08/2023
10066                                           mov     ax, [si+8]
10067                                           add     ax, [si+11h]
10068
10069                                           jnb     short cfb_retit
10070                                           jnz     short cfb_retit
10071                                                                   ; if carry set and ax=0
10072                                           dec     word [si+8]
10073                                           ;dec    word [cs:si+8] ; 0 -> 0FFFFh
10074                                                                   ; then decrease       BPB_TOTALSECTORS by 1
10075                                   %endif
10076                                           ; 18/12/2023
10077                                           ;cmp    word [bx+19], 0
10078 0000274F 8B4713                           mov     ax, [bx+19]    ; [bx+EBPB.TOTALSECTORS]
10079 00002752 21C0                             and     ax, ax ; 0 ?
10080 00002754 7412                             jz      short cfb_retit
10081
10082                                           ;mov    ax, [bx+19]
10083 00002756 03471C                           add     ax, [bx+28]    ; [bx+EBPB.HIDDENSECTORS]
10084 00002759 730D                             jnc     short cfb_retit
10085 0000275B 750B                             jnz     short cfb_retit
10086                                           ; ax = 0              ; 0 -> 0FFFFh
10087 0000275D FF4F13                           dec     word [bx+19]   ; then decrease       BPB_TOTALSECTORS by 1
10088
10089 00002760 836D1B01                         sub     word [di+1Bh], 1 ; [di+BDS.totalsecs32]
10090 00002764 835D1D00                         sbb     word [di+1Dh], 0 ; [di+BDS.totalsecs32+2]
10091                                   cfb_retit:
10092                                           ; 18/12/2023
10093                                           ;pop    si
10094                                           ;pop    dx
10095                                           ;pop    ax
10096
10097 00002768 C3                               retn
10098
10099                                   ; ---------------------------------------------------------------------------
10100
10101                                           ; 18/12/2023 - Retro DOS v5.0
10102                                           ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2A3Dh)
10103                                           ; ((MSDOS 6.22 IO.SYS - BIOSDATA:21DCh))
10104
10105 00002769 0200                     word2:          dw 2
10106 0000276B 0300                     word3:          dw 3
10107 0000276D 0002                     word512:    dw 512
10108
10109                                   ; =============== S U B   R O U T I N E =======================================
10110
10111                                   ; 15/10/2022
10112
10113                                   ; setdrvparms sets up the recommended bpb in each bds in the system based on
10114                                   ; the form factor. it is assumed that the bpbs for the various form factors
10115                                   ; are present in the bpbtable. for hard files, the recommended bpb is the same
10116                                   ; as the bpb on the drive.
10117                                   ;
10118                                   ; no attempt is made to preserve registers since we are going to jump to
10119                                   ; sysinit straight after this routine.
10120
10121                                           ; 18/12/2023 - Retro DOS v5.0
10122                                           ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2A43h)
10123                                   setdrvparms:
10124                                           ; 12/12/2023
10125                                           ; ds = cs
10126 0000276F 31DB                             xor     bx, bx
10127                                           ; 18/10/2022
10128 00002771 C43E[1901]                       les     di, [start_bds] ; get first bds in list
10129                                   _next_bds:
10130 00002775 06                               push    es
10131 00002776 57                               push    di
10132
10133                                           ; 18/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
10134 00002777 268A5D3E                         mov     bl, [es:di+3Eh]        ; [es:di+BDS.formfactor]
10135                                           ;mov    bl, [es:di+22h]        ; [es:di+BDS.formfactor]
10136
10137 0000277B 80FB05                           cmp     bl, 5          ; ffHardFile
10138 0000277E 753A                             jnz     short nothardff
10139 00002780 31D2                             xor     dx, dx
10140 00002782 268B450E                         mov     ax, [es:di+0Eh]        ; [es:di+BDS.totalsecs16]
10141 00002786 09C0                             or      ax, ax
10142 00002788 7508                             jnz     short get_ccyl
10143 0000278A 268B551D                         mov     dx, [es:di+1Dh]        ; [es:di+BDS.totalsecs32+2]
10144 0000278E 268B451B                         mov     ax, [es:di+1Bh]        ; [es:di+BDS.totalsecs32]
10145                                   get_ccyl:
10146 00002792 52                               push    dx
10147 00002793 50                               push    ax
10148 00002794 268B4515                         mov     ax, [es:di+15h]        ; [es:di+BDS.heads]
10149 00002798 26F76513                         mul     word [es:di+13h] ; [es:di+BDS.secpertrack]
10150                                                                   ; assume sectors per cyl. < 64k.
10151 0000279C 89C1                             mov     cx, ax         ; cx has # sectors per cylinder
10152 0000279E 58                               pop     ax
10153 0000279F 5A                               pop     dx             ; dx:ax = total sectors
10154 000027A0 50                               push    ax
10155 000027A1 89D0                             mov     ax, dx
10156 000027A3 31D2                             xor     dx, dx
10157 000027A5 F7F1                             div     cx
10158                                           ; 12/12/2023  ; !*!
10159                                           ; (data segment may not be same with code segment here)
10160                                           ;mov    [cs:temp_h], ax        ; ax be 0 here.
10161                                           ; 18/12/2023 - Retro DOS v5.0
10162                                           ;mov    [cs:saved_word], ax
10163 000027A7 58                               pop     ax
10164 000027A8 F7F1                             div     cx             ; div #sec by sec/cyl to get # cyl.
10165 000027AA 09D2                             or      dx, dx
10166 000027AC 7401                             jz      short no_cyl_rnd ; came out even
10167 000027AE 40                               inc     ax             ; round up
10168                                   no_cyl_rnd:
10169                                           ; 18/12/2023 - Retro DOS v5.0
10170 000027AF 26894541                         mov     [es:di+41h], ax        ; [es:di+BDS.cylinders]
```

```
10171                                           ;mov    [es:di+25h], ax       ; [es:di+BDS.cylinders]
10172
10173 000027B3 06                               push    es
10174 000027B4 1F                               pop     ds  ; !*! ; 12/12/2023
10175
10176 000027B5 8D7506                           lea     si, [di+6]      ; [di+BDS.bytespersec]
10177                                                                   ; ds:si -> bpb for hard       file
10178 000027B8 EB55                             jmp     short set_recbpb
10179                                   ; ---------------------------------------------------------------------------
10180
10181                                   nothardff:
10182 000027BA 0E                               push    cs
10183 000027BB 1F                               pop     ds
10184
10185                                   ; if fake floppy drive variable is set then we don't have to handle this bds.
10186                                   ; we can just go and deal with the next bds at label go_to_next_bds.
10187
10188                                           ; 10/12/2022
10189                                           ; ds = cs
10190                                           ; 17/10/2022 (ds=cs)
10191 000027BC 803E[131A]01                     cmp     byte [fakefloppydrv], 1
10192                                           ;cmp    byte [cs:fakefloppydrv], 1
10193 000027C1 7454                             jz      short go_to_next_bds
10194 000027C3 80FB07                           cmp     bl, 7           ; ffother
10195                                                                   ; special case "other" type of medium
10196 000027C6 753D                             jnz     short not_process_other
10197                                   process_other:
10198 000027C8 31D2                             xor     dx, dx
10199
10200                                           ;mov    ax, [di+25h]    ; [di+BDS.cylinders]
10201                                           ;mul    word [di+36h]   ; [di+BDS.rheads]
10202                                           ;mul    word [di+34h]   ; [di+BDS.rsecpertrack]
10203                                           ;mov    [di+2Fh], ax    ; [di+BDS.rtotalsecs16]
10204                                           ;                       ; have the total number of sectors
10205                                           ; 18/12/2023 - Retro DOS v5.0
10206 000027CA 8B4541                          mov     ax, [di+41h]    ; [di+BDS.cylinders]
10207 000027CD F76552                          mul     word [di+52h]   ; [di+BDS.rheads]
10208 000027D0 F76550                          mul     word [di+50h]   ; [di+BDS.rsecpertrack]
10209 000027D3 89454B                          mov     [di+4Bh], ax    ; [di+BDS.rtotalsecs16]
10210                                                                   ; have the total number of sectors
10211 000027D6 48                              dec     ax
10212 000027D7 B201                            mov     dl, 1
10213                                   _again:
10214 000027D9 3DF60F                          cmp     ax, 0FF6h       ; 4096-10
10215 000027DC 7206                            jb      short _@@
10216 000027DE D1E8                            shr     ax, 1
10217 000027E0 D0E2                            shl     dl, 1
10218 000027E2 EBF5                            jmp     short _again
10219                                   ; ---------------------------------------------------------------------------
10220
10221                                   _@@:
10222 000027E4 80FA01                          cmp     dl, 1           ; is it a small disk ?
10223 000027E7 7405                            jz      short __@@      ; yes, 224 root       entries is enuf
10224
10225                                           ; 18/12/2023 - Retro DOS v5.0
10226 000027E9 C74549F000                      mov     word [di+49h], 240 ; [di+BDS.rdirentries]
10227                                           ;mov    word [di+2Dh], 240 ; [di+BDS.rdirentries]
10228                                   __@@:
10229                                           ; 18/12/2023 - Retro DOS v5.0
10230 000027EE 885545                          mov     [di+45h], dl    ; [di+BDS.rsecperclus]
10231                                           ;mov    [di+29h], dl    ; [di+BDS.rsecperclus]
10232
10233                                   ; logic to get the sectors/fat area.
10234                                   ; fat entry is assumed to be 1.5 bytes!!!
10235
10236                                           ; 10/12/2022
10237                                           ; ds = cs
10238                                           ; 17/10/2022 (ds=cs)
10239 000027F1 F726[6B27]                       mul     word [word3]    ; * 3
10240 000027F5 F736[6927]                       div     word [word2]    ; / 2
10241 000027F9 31D2                             xor     dx, dx
10242 000027FB F736[6D27]                       div     word [word512] ; / 512
10243                                           ;
10244                                           ; 10/12/2022
10245                                           ;mul    word [cs:word3]       ; * 3
10246                                           ;div    word [cs:word2]       ; / 2
10247                                           ;xor    dx, dx
10248                                           ;div    word [cs:word512] ; / 512
10249                                           ;
10250 000027FF 40                              inc     ax              ; + 1
10251                                   no_round_up:
10252                                           ; 18/12/2023 - Retro DOS v5.0
10253 00002800 89454E                          mov     [di+4Eh], ax    ; [di+BDS.rfatsecs]
10254                                           ;mov    [di+32h], ax    ; [di+BDS.rfatsecs]
10255
10256 00002803 EB12                             jmp     short go_to_next_bds
10257                                   ; ---------------------------------------------------------------------------
10258
10259                                   not_process_other:
10260 00002805 D1E3                             shl     bx, 1           ; bx is word index into       table of bpbs
10261
10262                                           ;mov    si, bpbtable
10263                                           ;mov    si, [bpbtable+bx] ; 15/10/2022
10264                                           ; 09/12/2022
10265                                           ;mov    si, BPBTABLE
10266                                           ;mov    si, [bx+si]     ; get address of bpb
10267                                           ; 10/12/2022
10268                                           ;mov    si, [BPBTABLE+bx]
10269                                           ; 13/12/2022
10270                                           ;mov    si, [SYSINITOFFSET+bpbtable+bx] ; wrong ! 14/08/2023
10271
10272                                           ; 14/08/2023
10273                                           SYSINIT_OFFSET equ (SYSINITSEG-DOSBIODATASEG<<4)
10274                                                                                   ; correct offset
10275 00002807 8BB7[6E99]                       mov     si, [bx+SYSINIT_OFFSET+bpbtable]
10276
10277                                           ; 18/12/2023
10278                                           ; si = address of the requested disk(ette) parameter block
10279                                           ;       ! as offset from SYSINIT segment !
10280
10281                                           ; 28/08/2023
10282 0000280B 81C64049                        add     si, SYSINIT_OFFSET
10283                                                   ; + displacement from BIOSDATA segment ; 18/12/2023
10284                                   set_recbpb:
10285                                           ; 18/12/2023
10286                                           ;lea    di, [di+27h]    ; [di+BDS.R_BPB]
10287                                           ;                       ; es:di -> recbpb
10288                                           ;mov    cx, 25          ; bpbx.size
10289                                           ;rep movsb              ; move (size bpbx) bytes
10290
10291                                           ; 18/12/2023 - Retro DOS v5.0
10292 0000280F 8D7D43                          lea     di, [di+43h]    ; [di+BDS.R_BPB]
10293                                                                   ; es:di -> recbpb
10294 00002812 B93500                          mov     cx, 53          ; bpbx.size
```

```
10295 00002815 F3A4                     rep movsb               ; move (size bpbx) byte
10296                                   go_to_next_bds:
10297 00002817 5F                       pop     di
10298 00002818 07                       pop     es              ; restore pointer to bds
10299 00002819 26C43D                   les     di, [es:di]     ; [es:di+BDS.link]
10300 0000281C 83FFFF                   cmp     di, 0FFFFh      ; -1
10301 0000281F 740A                     jz      short got_end_of_bds_chain
10302 00002821 E951FF                   jmp     _next_bds
10303
10304                                   ; ---------------------------------------------------------------------------
10305
10306                                   ; 18/12/2022
10307                                   ;got_end_of_bds_chain:
10308                                   ;retn
10309
10310                                   ; =============== S U B   R O U T I N E =======================================
10311
10312                                   ; 15/10/2022
10313                                   ; 30/12/2018 - Retro DOS v4.0
10314
10315                                   ; al = device number
10316
10317                                   print_init:
10318 00002824 98                       cbw
10319 00002825 89C2                     mov     dx, ax
10320 00002827 B401                     mov     ah, 1
10321 00002829 CD17                     int     17h             ; PRINTER - INITIALIZE
10322                                                           ; DX = printer port (0-3)
10323                                                           ; Return: AH = status
10324                                   got_end_of_bds_chain:     ; 18/12/2022
10325 0000282B C3                       retn
10326
10327                                   ; =============== S U B   R O U T I N E =======================================
10328
10329                                   ; al = device number
10330
10331                                   aux_init:
10332 0000282C 98                       cbw
10333 0000282D 89C2                     mov     dx, ax
10334                                   ;mov    al, 0A3h        ; RSINIT ; 0A3h
10335                                                           ; 2400,n,1,8 (msequ.inc)
10336                                   ;mov    ah, 0
10337                                   ; 10/12/2022
10338 0000282F B8A300                   mov     ax, 00A3h
10339 00002832 CD14                     int     14h             ; SERIAL I/O - INITIALIZE USART
10340                                                           ;      AL = initializing parameters,
10341                                                           ;      DX = port number (0-3)
10342                                                           ; Return: AH = RS-232 status code bits,
10343                                                           ;         AL = modem status bits
10344 00002834 C3                       retn
10345
10346                                 ; =============== S U B   R O U T I N E =======================================
10347
10348                                 ; 18/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
10349                                 ; 08/08/2023 - Retro DOS v4.2 (Modified MSDOS 6.22 IO.SYS)
10350                                 ; 15/10/2022 (Modified MSDOS 5.0 IO.SYS) -Retro DOS v4 2022- (MSDOS 5.0-6.21)
10351                                 ; 30/12/2018 - Retro DOS v4.0
10352                                 ; 03/06/2018 - Retro DOS v3.0
10353                                 ; (19/03/2018 - Retro DOS v2.0)
10354
10355                                 ; domini ******************************************************************************
10356                                 ;
10357                                 ;mini disk initialization routine. called right after dohard
10358                                 ;modified for >2 hardfile support
10359                                 ;
10360                                 ; **cs=ds=es=datagrp
10361                                 ;
10362                                 ; **domini will search for every extended partition in the system, and
10363                                 ;   initialize it.
10364                                 ;
10365                                 ; **bdsm stands for bds table for mini disk and located right after the label
10366                                 ;   end96tpi. end_of_bdsm will have the offset value of the ending
10367                                 ;   address of bdsm table.
10368                                 ;
10369                                 ; **bdsm is the same as usual bds structure except that tim_lo, tim_hi entries
10370                                 ;   are overlapped and used to identify mini disk and the number of hidden_trks.
10371                                 ;   right now, they are called as ismini, hidden_trks respectively.
10372                                 ;
10373                                 ; **domini will use the same routine in sethard routine after label set2 to
10374                                 ;   save coding.
10375                                 ;
10376                                 ; **drvmax determined in dohard routine will be used for the next
10377                                 ;   available logical mini disk drive number.
10378                                 ;
10379                                 ; input: drvmax, dskdrvs
10380                                 ;
10381                                 ; output: minidisk installed. bdsm table established and installed to bds.
10382                                 ;     end_of_bdsm - ending offset address of bdsm.
10383                                 ;
10384                                 ; called modules:
10385                                 ;             getboot
10386                                 ;             find_mini_partition (new), xinstall_bds (new), M038
10387                                 ;
10388                                 ;             setmini (new, it will use set2 routine)
10389                                 ;
10390                                 ; variables used: end_of_bdsm
10391                                 ;             rom_minidisk_num
10392                                 ;             mini_hdlim, mini_seclim
10393                                 ;             BDS_STRUC, start_bds
10394                                 ;
10395                                 ;******************************************************************************
10396
10397                                             ; 18/12/2023 - Retro DOS v5.0 IO.SYS/IBMBIO.COM
10398                                             ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2B10h)
10399
10400                                             ; 19/10/2022
10401                                 domini:
10402 00002835 8A36[5F1A]                       mov     dh, [hnum]      ; get number of hardfiles
10403                                             ; 10/12/2022
10404 00002839 20F6                             and     dh, dh
10405                                           ;cmp    dh, 0
10406 0000283B 743C                             jz      short dominiret         ; no hard file?       then exit.
10407 0000283D B280                             mov     dl, 80h         ; start with hardfile 80h
10408                                 domini_loop:
10409                                             ; 18/12/2023 - Retro DOS v5.0
10410 0000283F 31C0                             xor     ax, ax ; 0
10411                                             ; ds = cs
10412                                             ;mov    [cs:ep_start_sector], ax
10413                                             ;mov    [cs:ep_start_sector+2], ax
10414                                             ;mov    [cs:ep_hidden_secs], ax
10415                                             ;mov    [cs:ep_hidden_secs+2], ax
10416 00002841 A3[8523]                         mov     [ep_start_sector], ax
10417 00002844 A3[8723]                         mov     [ep_start_sector+2], ax
10418 00002847 A3[8923]                         mov     [ep_hidden_secs], ax
```

```
10419 0000284A A3[8B23]                         mov     [ep_hidden_secs+2], ax
10420                                           ;
10421 0000284D 52                               push    dx
10422 0000284E 8816[5E1A]                       mov     [rom_minidisk_num], dl
10423 00002852 B408                             mov     ah, 8
10424 00002854 CD13                             int     13h             ; DISK - DISK -       GET CURRENT DRIVE PARAMETERS (XT,AT,XT286,CONV,PS)
10425                                                                   ; DL = drive number
10426                                                                   ; Return: CF set on error, AH =      status code, BL       = drive type
10427                                                                   ; DL = number of consecutive drives
10428                                                                   ; DH = maximum value for head number, ES:DI -> drive parameter
10429
10430                                           ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
10431                                           ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:2B36h
10432                                           ;inc    dh
10433                                           ;xor    ax, ax
10434                                           ;mov    al, dh
10435 00002856 31C0                             xor     ax, ax
10436 00002858 88F0                             mov     al, dh  ; <= 255
10437 0000285A 40                               inc     ax      ; (0FFh -> 100h)
10438 0000285B A3[641A]                         mov     [mini_hdlim], ax ; # of heads
10439                                           ;and    cl, 3Fh
10440                                           ;mov    al, cl
10441                                           ; 08/08/2023
10442 0000285E 88C8                             mov     al, cl
10443 00002860 83E03F                           and     ax, 3Fh
10444 00002863 A3[661A]                         mov     [mini_seclim], ax ; # of sectors/track
10445
10446                                           ; 18/12/2023
10447                                           ;push   es ; * ; not necessary
10448 00002866 8A16[5E1A]                       mov     dl, [rom_minidisk_num]
10449 0000286A E860FA                           call    getboot         ; read master boot record into
10450                                                                   ; initbootsegment:bootbias
10451 0000286D 7203                             jc      short domininext
10452 0000286F E80800                           call    find_mini_partition
10453                                   domininext:
10454                                           ;pop    es ; *
10455 00002872 5A                               pop     dx
10456 00002873 FEC2                             inc     dl              ; next hard file
10457 00002875 FECE                             dec     dh
10458 00002877 75C6                             jnz     short domini_loop
10459                                   dominiret:
10460 00002879 C3                               retn
10461
10462                                   ; =============== S U B   R O U T I N E =======================================
10463
10464                                   ; 15/10/2022 (Modified MSDOS 5.0 IO.SYS)
10465                                   ; 30/12/2018 - Retro DOS v4.0
10466
10467                                   ;find_mini_partition tries to find every extended partition on a disk.
10468                                   ;at entry:  di -> bdsm entry
10469                                   ;           es:bx -> 07c0:bootbias - master boot record
10470                                   ;           rom_minidisk_num - rom drive number
10471                                   ;           drvmax - logical drive number
10472                                   ;           mini_hdlim, mini_seclim
10473                                   ;
10474                                   ;called routine: setmini which uses set2 (in sethard routine)
10475                                   ;variables & equates used from original bios - flags, fnon_removable, fbigfat
10476
10477                                           ; 19/12/2023 - Retro DOS v5.0
10478                                           ;       (Modified PCDOS 7.1 IBMBIO.COM)
10479                                           ; (PCDOS 7.1 IBMBIO.COM - BIOSADATA:2BFCh)
10480
10481                                   find_mini_partition:
10482 0000287A 81C3C201                         add     bx, 1C2h        ; bx -> file system id
10483
10484                                           ; 19/12/2023
10485                                           ; PCDOS 7.1 IBMBIO.COM
10486                                           ;mov    word [ld_p_number], 26
10487                                   fmpnext:
10488                                           ;add    word [ld_p_number], 16
10489                                           ;cmp    word [ld_p_number], 4122
10490                                           ;               ; 64 logical disk partitions (64 EBRs)
10491                                           ;               ; (64*4 = 256 pte's, 256*16 = 4096, + 26 = 4122)
10492                                           ;jg     short fmpnextfound
10493
10494 0000287E 26803F05                         cmp     byte [es:bx], 5 ; 05h = extended partition id.
10495 00002882 7410                             je      short fmpgot ; Extended DOS CHS
10496
10497                                           ; 19/12/2023 - Retro DOS v5.0
10498 00002884 26803F0F                         cmp     byte [es:bx], 0Fh ; Extended DOS LBA
10499 00002888 740A                             je      short fmpgot
10500
10501 0000288A 83C310                           add     bx, 16
10502 0000288D 81FB0204                         cmp     bx, 402h        ; 202h+bootbias
10503 00002891 75EB                             jne     short fmpnext
10504                                           ;jmp    short fmpnextfound ; extended partition     not found
10505                                           ; 18/12/2022
10506                                   fmpnextfound:
10507 00002893 C3                               retn
10508
10509                                   ;       ; 30/07/2019 - Retro DOS v3.2
10510                                   ;       jb      short fmpnext
10511                                   ;fmpret:
10512                                   ;       retn    ; 29/05/2019
10513
10514                                   ; ---------------------------------------------------------------------------
10515
10516                                           ; 19/10/2022
10517                                   fmpgot:                                 ; found my partition.
10518 00002894 E82B01                           call    dmax_check      ; check for drvmax already 26
10519 00002897 73FA                             jnb     short fmpnextfound ; done if too many
10520
10521 00002899 8B3E[621A]                       mov     di, [end_of_bdss] ; get next free bds
10522
10523                                           ; 19/12/2023
10524                                           ;mov    word [di+47h], 1 ; [di+BDS.bdsm_ismini]
10525                                           ;; 10/12/2022
10526                                           ;or     byte [di+23h], 1
10527                                           ;;or    word [di+23h], 1 ; [di+BDS.flags]
10528                                           ;                       ; fNon_Removable
10529                                           ;mov    byte [di+22h], 5 ; [di+BDS.formfactor]
10530                                           ;                       ; ffHardFile
10531                                           ; 19/12/2023 - Retro DOS v5.0
10532 0000289D C745790100                       mov     word [di+79h], 1 ; [di+BDS.bdsm_ismini]
10533 000028A2 804D3F01                         or      byte [di+3Fh], 1 ; [di+BDS.flags], fNon_Removable
10534 000028A6 C6453E05                         mov     byte [di+3Eh], 5 ; [di+BDS.formfactor], ffHardFile
10535
10536 000028AA C606[081A]00                     mov     byte [fbigfat], 0 ; assume 12 bit fat.
10537 000028AF A1[641A]                         mov     ax, [mini_hdlim]
10538 000028B2 894515                           mov     [di+15h], ax    ; [di+BDS.heads]
10539 000028B5 A1[661A]                         mov     ax, [mini_seclim]
10540 000028B8 894513                           mov     [di+13h], ax    ; [di+BDS.secpertrack]
10541 000028BB A0[5E1A]                         mov     al, [rom_minidisk_num]
10542 000028BE 884504                           mov     [di+4], al      ; [di+BDS.drivenum]
```

```
10543                                                   ; set physical number
10544 000028C1 A0[7500]                         mov     al, [drvmax]
10545 000028C4 884505                           mov     [di+5], al      ; [di+BDS.drivelet]
10546                                                   ; set logical number
10547 000028C7 26837F0A00                       cmp     word [es:bx+10], 0
10548                                           ;ja     short fmpgot_cont
10549 000028CC 7707                             ja      short fmpgot1  ; 19/12/2023
10550 000028CE 26837F0840                       cmp     word [es:bx+8], 64 ; with current bpb,
10551                                                   ; only lower word is meaningful.
10552 000028D3 72BE                             jb      short fmpnextfound
10553                                                   ; should be bigger than 64 sectors at least
10554                                   fmpgot1:    ; 19/12/2023
10555                                   ;fmpgot_cont:
10556 000028D5 83EB04                           sub     bx, 4           ; let bx point to the start of the entry
10557 000028D8 268A7702                         mov     dh, [es:bx+2]  ; cylinder
10558 000028DC 80E6C0                           and     dh, 0C0h        ; get higher bits of cyl
10559 000028DF D0C6                             rol     dh, 1
10560 000028E1 D0C6                             rol     dh, 1
10561 000028E3 268A5703                         mov     dl, [es:bx+3]  ; cyl byte
10562                                           ; 19/12/2023 - Retro DOS v5.0
10563 000028E7 89557B                           mov     [di+7Bh], dx   ; [di+BDS.bdsm_hidden_trks]
10564                                           ;mov    [di+49h], dx   ; [di+BDS.bdsm_hidden_trks]
10565                                                   ; set hidden trks
10566                                           ; 19/12/2023
10567                                           ;push   bx ; * ; PCDOS 7.1
10568                                           ;;;
10569 000028EA 268B4F08                         mov     cx, [es:bx+8]  ; partition size, lw
10570 000028EE 268B470A                         mov     ax, [es:bx+10] ; partition size, hw
10571 000028F2 030E[8523]                       add     cx, [ep_start_sector]
10572 000028F6 1306[8723]                       adc     ax, [ep_start_sector+2]
10573 000028FA 31D2                             xor     dx, dx ; 19/12/2023
10574 000028FC 3916[8523]                       cmp     [ep_start_sector], dx ; 0
10575                                           ;cmp    word [ep_start_sector], 0
10576 00002900 750D                             jnz     short fmpgot2
10577 00002902 3916[8723]                       cmp     [ep_start_sector+2], dx ; 0
10578                                           ;cmp    word [ep_start_sector+2], 0
10579 00002906 7507                             jnz     short fmpgot2
10580 00002908 890E[8523]                       mov     [ep_start_sector], cx
10581 0000290C A3[8723]                         mov     [ep_start_sector+2], ax
10582                                   fmpgot2:
10583 0000290F 890E[8923]                       mov     [ep_hidden_secs], cx
10584 00002913 A3[8B23]                         mov     [ep_hidden_secs+2], ax
10585
10586                                           ; convert start sector address to CHS
10587
10588                                           ; 19/12/2023
10589                                           ; dx = 0
10590                                           ;push   bx ; * ; not necessary
10591
10592                                           ;mov    bx, [di+13h]   ; [di+BDS.secpertrack]
10593 00002916 8B7513                           mov     si, [di+13h]   ; [di+BDS.secpertrack]
10594                                           ;xor    dx, dx  ; dx = 0
10595                                           ;div    bx
10596 00002919 F7F6                             div     si
10597 0000291B 91                               xchg    ax, cx
10598                                           ;div    bx
10599 0000291C F7F6                             div     si
10600                                           ;mov    bx, [di+15h]   ; [di+BDS.heads]
10601                                           ; 17/04/2024 (BugFix)
10602 0000291E 8B7515                           mov     si, [di+15h]   ; [di+BDS.heads]
10603 00002921 91                               xchg    ax, cx
10604 00002922 31D2                             xor     dx, dx
10605                                           ;div    bx
10606 00002924 F7F6                             div     si
10607 00002926 91                               xchg    ax, cx
10608                                           ;div    bx
10609 00002927 F7F6                             div     si
10610
10611                                           ;pop    bx ; *
10612
10613 00002929 09C9                             or      cx, cx
10614 0000292B 7505                             jnz     short fmpgot_lba_rd
10615 0000292D 3D0004                           cmp     ax, 1024        ; cylinder number < 1024, CHS read is proper
10616 00002930 7235                             jb      short fmpgot_chs_rd
10617                                   fmpgot_lba_rd:
10618 00002932 804D4004                         or      byte [di+40h], 4 ; set fLBArw flag ; LBA read/write ok/ready
10619 00002936 8A16[5E1A]                       mov     dl, [rom_minidisk_num]
10620 0000293A 1E                               push    ds
10621                                           ; 19/12/2023
10622                                           ;push   si ; ** ; not necessary
10623 0000293B 31C0                             xor     ax, ax          ; push bp
10624                                                                   ; mov bp, sp ; (*)
10625 0000293D 50                               push    ax ; 0
10626 0000293E 50                               push    ax ; 0
10627 0000293F FF36[8B23]                       push    word [ep_hidden_secs+2]
10628 00002943 FF36[8923]                       push    word [ep_hidden_secs]
10629 00002947 B80002                           mov     ax, bootbias ; 200h
10630                                           ;mov    ax, 200h        ; bootbias (buffer offset)
10631 0000294A 06                               push    es              ; buffer segment
10632 0000294B 50                               push    ax
10633 0000294C B80100                           mov     ax, 1
10634 0000294F 50                               push    ax              ; read count
10635 00002950 B81000                           mov     ax, 10h         ; DAP size = 16
10636 00002953 50                               push    ax
10637 00002954 8CD0                             mov     ax, ss
10638 00002956 8ED8                             mov     ds, ax
10639 00002958 89E6                             mov     si, sp          ; ds:si = Disk Address Packet
10640
10641 0000295A B442                             mov     ah, 42h         ; LBA read
10642 0000295C CD13                             int     13h             ; DISK - IBM/MS Extension
10643                                                                   ; EXTENDED READ (DL - drive, DS:SI - disk address packet)
10644                                           ; 19/12/2023
10645                                           ;pushf          ; PCDOS 7.1 IBMBIO.COM BUG! Erdogan Tan - 08/08/2023
10646                                           ;add    sp, 16
10647                                           ;popf           ; BUG!
10648                                                                   ; mov sp, bp ; (*)
10649                                                                   ; pop bp
10650                                           ; 19/12/2023
10651 0000295E 9F                               lahf            ; load status flags into AH
10652 0000295F 83C410                           add     sp, 16
10653 00002962 9E                               sahf            ; store AH into flags
10654
10655                                           ;pop    si ; ** ; 19/12/2023
10656 00002963 1F                               pop     ds
10657 00002964 7317                             jnc     short fmpgot3
10658                                   fmpnotfound:        ; 19/12/2023
10659 00002966 C3                               retn
10660                                           ;jmp    short fmpgot3
10661                                           ;;;
10662
10663                                           ; 19/12/2023
10664                                   fmpgot_chs_rd:
10665 00002967 268B4F02                         mov     cx, [es:bx+2]  ; cylinder,cylinder/sector
10666 0000296B 268A7701                         mov     dh, [es:bx+1]  ; head
```

```
10667 0000296F 8A16[5E1A]                      mov     dl, [rom_minidisk_num]
10668 00002973 BB0002                          mov     bx, 200h        ; bootbias
10669 00002976 B80102                          mov     ax, 201h
10670 00002979 CD13                            int     13h             ; DISK - READ SECTORS INTO MEMORY
10671                                                                  ; AL = number of sectors to read, CH = track, CL = sector
10672                                                                  ; DH = head, DL       = drive, ES:BX -> buffer to fill
10673                                                                  ; Return: CF set on error, AH =      status, AL = number of sectors read
10674                                  ;fmpgot3:      ; 19/12/2023
10675                                                  ;jc     short fmpnextfound
10676 0000297B 72E9                              jc      short fmpnotfound
10677                                  fmpgot3:
10678 0000297D BBC203                            mov     bx, 3C2h        ; 1C2h+bootbias
10679
10680                                                  ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
10681                                                  ; PCDOS 7.1 IBMBIO.COM - BIOSDATA:2C7Ch
10682 00002980 26817F3C55AA                      cmp     word [es:bx+3Ch], 0AA55h ; 03C2h+03Ch = 3FEh
10683                                                  ;jne    short fmpnextfound ; not a valid boot sector !
10684                                                  ; 19/12/2023
10685 00002986 75DE                              jne     short fmpnotfound ; not a valid boot sector !
10686
10687                                                  ; 13/08/2023
10688                                                  ;push   es
10689 00002988 E80800                            call    setmini         ; install a mini disk.
10690                                                                    ; bx value saved.
10691                                                  ;pop    es  ; 13/08/2023
10692 0000298B 7203                              jc      short fmpnextchain
10693 0000298D E84700                            call    xinstall_bds    ; -- install the bdsm into table
10694                                  fmpnextchain:
10695 00002990 E9EBFE                            jmp     fmpnext         ; let's find out
10696                                                                    ; if we have any chained partition
10697                                  ; ---------------------------------------------------------------------------
10698
10699                                                  ; 18/12/2022
10700                                  ;fmpnextfound:
10701                                                  ;retn
10702
10703                                  ; =============== S U B   R O U T I N E =======================================
10704
10705                                  ; 15/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
10706                                  ; 28/12/2018 - Retro DOS v4.0 (MSDOS 6.21)
10707
10708                                  ; 19/12/2022 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
10709                                  ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2C92h)
10710
10711                                  setmini:    ; 'setmini' is called from 'find_mini_partition' procedure
10712
10713 00002993 57                                 push    di
10714 00002994 53                                 push    bx
10715                                             ; 12/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
10716                                             ; ds = cs = BIOSDATA segment
10717                                             ;push   ds
10718 00002995 06                                 push    es
10719                                  setmini_1:
10720                                             ;cmp    byte [es:bx], 1 ; FAT12 partition
10721                                             ;je     short setmini_2
10722                                             ;cmp    byte [es:bx], 4 ; FAT16 (CHS) partition
10723                                             ;je     short setmini_2
10724                                             ;cmp    byte [es:bx], 6 ; FAT16 BIG (CHS) partition
10725                                             ;je     short setmini_2
10726                                             ;
10727                                             ; 19/12/2023 - Retro DOS v5.0
10728                                             ;cmp    byte [es:bx], 0Bh ; FAT32 (CHS) partition
10729                                             ;je     short setmini_2
10730                                             ;cmp    byte [es:bx], 0Ch ; FAT32 (LBA) partition
10731                                             ;je     short setmini_2
10732                                             ;cmp    byte [es:bx], 0Eh ; FAT16 (LBA) partition
10733                                             ;je     short setmini_2
10734
10735                                             ; 19/12/2023
10736 00002996 268A07                            mov     al, [es:bx]
10737 00002999 3C01                               cmp     al, 1           ; FAT12 partition
10738 0000299B 7422                               je      short setmini_2
10739 0000299D 3C04                               cmp     al, 4           ; FAT16 (CHS) partition
10740 0000299F 741E                               je      short setmini_2
10741 000029A1 3C06                               cmp     al, 6           ; FAT16 BIG (CHS) partition
10742 000029A3 741A                               je      short setmini_2
10743 000029A5 3C0B                               cmp     al, 0Bh         ; FAT32 (CHS) partition
10744 000029A7 7416                               je      short setmini_2
10745 000029A9 3C0C                               cmp     al, 0Ch         ; FAT32 (LBA) partition
10746 000029AB 7412                               je      short setmini_2
10747 000029AD 3C0E                               cmp     al, 0Eh         ; FAT16 (LBA) partition
10748 000029AF 740E                               je      short setmini_2
10749
10750 000029B1 83C310                             add     bx, 16
10751 000029B4 81FB0204                           cmp     bx, 402h        ; 202h+bootbias
10752                                             ;jne    short setmini_1
10753 000029B8 72DC                               jb      short setmini_1 ; 19/12/2023
10754 000029BA F9                                 stc
10755 000029BB 07                                 pop     es
10756                                             ; 12/08/2023
10757                                             ;pop    ds
10758 000029BC 5B                                 pop     bx
10759 000029BD 5F                                 pop     di
10760 000029BE C3                                 retn
10761
10762                                  ; ---------------------------------------------------------------------------
10763                                  setmini_2:
10764 000029BF E9CBF9                             jmp     set2            ; branch into middle of sethard
10765
10766                                  ; =============== S U B   R O U T I N E =======================================
10767
10768                                  ; 30/12/2022 - Retro DOS v4.2
10769                                  ; (SYSINITSEG is 473h for MSDOS 6.21 IO.SYS)
10770
10771                                  ; 15/10/2022
10772                                  ; 28/12/2018 - Retro DOS v4.0
10773                                  ;
10774                                  ; dmax_check -- call this when we want to install a new drive.
10775                                  ;           it checks for drvmax < 26 to see if there is
10776                                  ;           a drive letter left.
10777                                  ;
10778                                  ;   drvmax < 26 : carry SET!
10779                                  ;   drvmax >=26 : carry RESET!, error flag set for message later
10780                                  ;                 trash ax
10781
10782                                             ; 19/12/2023 - Retro DOS v5.0
10783                                  dmax_check:
10784 000029C2 803E[7500]1A                       cmp     byte [drvmax], 26 ; checks for drvmax < 26
10785 000029C7 720D                               jb      short dmax_ok  ; return with carry if okay
10786 000029C9 06                                 push    es
10787                                             ;;mov   ax, 46Dh        ; SYSINIT_SEG (SYSINIT segment)
10788                                             ;mov    ax, 544h        ; 19/12/2023 (PCDOS 7.1)
10789 000029CA B80405                             mov     ax, SYSINITSEG ; 17/10/2022
10790 000029CD 8EC0                               mov     es, ax
```

```
                                        ; 18/10/2022
000029CF 26C606[8803]01                 mov     byte [es:TOOMANYDRIVESFLAG], 1 ; 09/12/2022
                                        ;mov    byte ptr es:3FFh, 1 ; [es:toomanydrivesflag]
                                                                ; set message flag
                                                                ; [SYSINIT+toomanydrivesflag]
000029D5 07                             pop     es

                                        ;;push  es
                                        ;;mov   ax,SYSINIT_SEG
                                        ;;mov   es,ax
                                        ;;mov   byte [es:toomanydrivesflag],1
                                                                ; set message flag
                                        ;;pop   es
                                        ;
                                        ;mov    byte [SYSINIT+toomanydrivesflag],1
                                dmax_ok:
000029D6 C3                             retn

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

                                ; 18/10/2022
                                ; 15/10/2022
                                ; 28/12/2018 - Retro DOS v4.0
                                ;
                                ;   link next bds (at ds:di) into the chain. assume that the
                                ;     chain is entirely within ds == datagrp. also update drvmax,
                                ;     dskdrv_table, and end_of_bdss.

                                        ; 19/12/2023 - Retro DOS v5.0
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSDATA:2CE1h)
                                xinstall_bds:
000029D7 56                             push    si
000029D8 53                             push    bx
000029D9 8B36[1901]                     mov     si, [start_bds]       ; get first bds
                                xinstall_bds_1:
000029DD 833CFF                         cmp     word [si], 0FFFFh ; is this the last one?
000029E0 7404                           jz      short xinstall_bds_2 ;skip ahead if so
                                        ;mov    si, [si+BDS.link]
000029E2 8B34                           mov     si, [si]        ; chain through list
000029E4 EBF7                           jmp     short xinstall_bds_1

                                xinstall_bds_2:
                                        ;mov    [si+BDS.link], di
000029E6 893C                           mov     [si], di
                                        ;mov    [si+BDS.link+2], ds
000029E8 8C5C02                         mov     [si+2], ds
                                        ;mov    word [di+BDS.link], -1
000029EB C705FFFF                       mov     word [di], 0FFFFh ; make sure it is a null ptr.
                                        ;mov    [di+BDS.link+2], ds
000029EF 8C5D02                         mov     [di+2], ds ; might as well plug segment
                                        ; 20/03/2019 - Retro DOS v4.0
                                        ;lea    bx, [di+BDS.BPB]
000029F2 8D5D06                         lea     bx, [di+6]
000029F5 8B36[601A]                     mov     si, [last_dskdrv_table]
000029F9 891C                           mov     [si], bx
000029FB 8306[601A]02                   add     word [last_dskdrv_table], 2
00002A00 FE06[7500]                     inc     byte [drvmax]
                                        ;add    word [end_of_bdss], 100 ; BDS.size = 100
                                        ; 19/12/2023 - Retro DOS v5.0
00002A04 8106[621A]9600                 add     word [end_of_bdss], 150 ; BDS.size = 150
00002A0A 5B                             pop     bx
00002A0B 5E                             pop     si
00002A0C C3                             retn

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

                                ; 17/10/2022
                                ; 15/10/2022
                                ; 28/12/2018 - Retro DOS v4.0
                                ; 03/06/2018 - Retro DOS v3.0

                                        ; 19/12/2023 - Retro DOS v5.0
                                cmos_clock_read:
00002A0D 50                             push    ax
00002A0E 51                             push    cx
00002A0F 52                             push    dx
00002A10 55                             push    bp
00002A11 31ED                           xor     bp, bp
                                loop_clock:
00002A13 31C9                           xor     cx, cx
00002A15 31D2                           xor     dx, dx
00002A17 B402                           mov     ah, 2
00002A19 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
                                        ; 19/12/2023
                                        ;cmp    cx, 0
00002A1B 21C9                           and     cx, cx
00002A1D 750F                           jnz     short clock_present
                                        ;cmp    dx, 0
00002A1F 09D2                           or      dx, dx
00002A21 750B                           jnz     short clock_present
                                        ;cmp    bp, 1           ; read again after a slight delay, in case clock
                                        ;je     short no_readdate ; was       at zero setting.
00002A23 21ED                           and     bp, bp
00002A25 751A                           jnz     short no_readdate
00002A27 45                             inc     bp              ; only perform delay once.
                                        ;mov    cx, 4000h       ; 16384
                                        ; 19/12/2023
00002A28 B540                           mov     ch, 40h ; cx = 4000h ; 16384
                                delay:
00002A2A E2FE                           loop    delay
00002A2C EBE5                           jmp     short loop_clock
                                ; ---------------------------------------------------------------------------

                                clock_present:
                                        ;mov    byte [cs:havecmosclock], 1 ; set the flag for cmos clock
                                        ; 19/12/2023
                                        ; ds = cs
00002A2E C606[8C04]01                   mov     byte [havecmosclock], 1 ; set the flag for cmos clock

00002A33 E81000                         call    cmosck          ; reset cmos clock rate       that may be
                                                                ; possibly destroyed by       cp dos and
                                                                ; post routine did not restore that.
00002A36 56                             push    si
00002A37 E831EE                         call    read_real_date ; read real-time clock for date
00002A3A FA                             cli
                                        ;mov    ds:daycnt, si  ; set system date
00002A3B 8936[8904]                     mov     [daycnt], si
00002A3F FB                             sti
00002A40 5E                             pop     si
                                no_readdate:
00002A41 5D                             pop     bp
```

```
10915 00002A42 5A                           pop     dx
10916 00002A43 59                           pop     cx
10917 00002A44 58                           pop     ax
10918                                cmosck9:    ; 19/12/2023
10919 00002A45 C3                           retn
10920
10921                                ; ---------------------------------------------------------------------------
10922
10923                                ; the following code is written by jack gulley in engineering group.
10924                                ; cp dos (CP/DOS, OS/2) is changing cmos clock rate for its own purposes
10925                                ; and if the use cold boot the system to use pc dos while running cp dos,
10926                                ; the cmos clock rate are still slow which slow down disk operations
10927                                ; of pc dos which uses cmos clock. pc dos is put this code in msinit
10928                                ; to fix this problem at the request of cp dos.
10929                                ;
10930                                ; the program is modified to be run on msinit. equates are defined
10931                                ; in cmosequ.inc. this program will be called by cmos_clock_read procedure.
10932                                ;
10933                                ;  the following code cmosck is used to insure that the cmos has not
10934                                ;   had its rate controls left in an invalid state on older at's.
10935                                ;
10936                                ;   it checks for an at model byte "fc" with a submodel type of
10937                                ;   00, 01, 02, 03 or 06 and resets the periodic interrupt rate
10938                                ;   bits in case post has not done it. this initialization routine
10939                                ;   is only needed once when dos loads. it should be run as soon
10940                                ;   as possible to prevent slow diskette access.
10941                                ;
10942                                ;   this code exposes one to dos clearing cmos setup done by a
10943                                ;   resident program that hides and re-boots the system.
10944
10945                                cmosck:                                 ; check and reset rtc rate bits
10946
10947                                ;model byte and submodel byte were already determined in msinit.
10948
10949                                    ; 16/06/2018 - Retro DOS v3.0
10950                                    ; 19/03/2018 (Model: 0FCh, Sub Model: 01h, REF: AMIBIOS Prog. Guide)
10951
10952                                    ; 19/12/2023 - Retro DOS v5.0
10953
10954                                    ; 19/12/2023
10955                                    ; ds = cs
10956                                    ;push   ax ; not necessary ; 19/12/2023
10957                                    ;
10958 00002A46 803E[AF05]FC               cmp     byte [model_byte], 0FCh
10959                                    ;cmp    byte [cs:model_byte], 0FCh
10960 00002A4B 75F8                     jnz     short cmosck9  ; Exit if not an AT model
10961 00002A4D 803E[B005]06             cmp     byte [secondary_model_byte], 6  ; 21/04/2024
10962                                    ;cmp    byte [cs:secondary_model_byte], 6
10963                                                    ; Is it 06 for the industral AT ?
10964 00002A52 7407                     jz      short cmosck4  ; Go reset CMOS periodic rate if 06
10965 00002A54 803E[B005]04             cmp     byte [secondary_model_byte], 4
10966                                    ;cmp    byte [cs:secondary_model_byte], 4
10967                                                    ; Is it 00, 01, 02, or 03 ?
10968 00002A59 73EA                     jnb     short cmosck9  ; EXIT if problem fixed by POST
10969                                                    ; Also,Secondary_model_byte = 0
10970                                                    ;   when AH=0C0h, int 15h failed.
10971                                                    ;       RESET THE CMOS PERIODIC RATE
10972                                                    ;  Model=FC submodel=00,01,02,03 or 06
10973                                cmosck4:
10974 00002A5B B08A                             mov     al, 8Ah         ; cmos_reg_a|nmi
10975                                                                     ; NMI disabled on return
10976 00002A5D B426                             mov     ah, 26h         ; 00100110b
10977                                                                     ; Set divider & rate selection
10978 00002A5F E80B00                           call    cmos_write
10979 00002A62 B08B                             mov     al, 8Bh         ; cmos_reg_b|nmi
10980                                                                     ; NMI disabled on return
10981 00002A64 E82000                           call    cmos_read
10982 00002A67 2407                             and     al, 7           ; 00000111b
10983                                                                     ; clear SET,PIE,AIE,UIE,SQWE
10984 00002A69 88C4                             mov     ah, al
10985 00002A6B B00B                             mov     al, 0Bh         ; cmos_reg_b
10986                                                                     ; NMI enabled on return
10987                                                   ; 19/12/2023
10988                                                   ;call   cmos_write
10989                                ;cmosck9:
10990                                                   ;pop    ax ; 19/12/2023
10991                                                   ;retn
10992
10993                                                   ; 19/12/2023
10994                                                   ;jmp    short cmos_write
10995
10996                                ; =============== S U B   R O U T I N E =======================================
10997
10998                                ;--- cmos_write ------------------------------------------------------------
10999                                ;           write byte to cmos system clock configuration table       :
11000                                ;                                                                     :
11001                                ; input: (al)=     cmos table address to be written to                    :
11002                                ;           bit    7 = 0 for nmi enabled and 1 for nmi disabled on exit   :
11003                                ;           bits 6-0 = address of table location to write                 :
11004                                ;     (ah)= new value to be placed in the addressed table location        :
11005                                ;                                                                     :
11006                                ; output:   value in (ah) placed in location (al) with nmi left disabled  :
11007                                ;           if bit 7 of (al) is on. during the cmos update both nmi and   :
11008                                ;           normal interrupts are disabled to protect cmos data integrity. :
11009                                ;           the cmos address register is pointed to a default value and   :
11010                                ;           the interrupt flag restored to the entry state on return.     :
11011                                ;           only the cmos location and the nmi state is changed.          :
11012                                ;---------------------------------------------------------------------------
11013
11014                                cmos_write:                             ; write (ah) to location (al)
11015 00002A6D 9C                               pushf                       ;
11016 00002A6E 50                               push    ax                  ; save work register values
11017 00002A6F FA                               cli
11018 00002A70 50                               push    ax                  ; save user nmi       state
11019 00002A71 0C80                             or      al, 80h             ; disable nmi for us
11020 00002A73 E670                             out     70h, al             ; CMOS Memory/RTC Index       Register:
11021                                                                       ; RTC Seconds
11022 00002A75 90                               nop
11023 00002A76 88E0                             mov     al, ah
11024 00002A78 E671                             out     71h, al             ; CMOS Memory/RTC Data Register
11025 00002A7A 58                               pop     ax                  ; get user nmi
11026 00002A7B 2480                             and     al, 80h
11027 00002A7D 0C0F                             or      al, 0Fh
11028 00002A7F E670                             out     70h, al             ; CMOS Memory/RTC Index       Register:
11029                                                                       ; RTC Seconds
11030 00002A81 90                               nop
11031 00002A82 E471                             in      al, 71h             ; CMOS Memory/RTC Data Register
11032 00002A84 58                               pop     ax                  ; restore work registers
11033
11034                                                   ; 19/12/2023
11035                                                   ;push   cs              ; *place code segment in stack and
11036                                                   ;call   cmos_popf       ; *handle popf for b- level 80286
11037                                                   ;retn
11038 00002A85 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)
00002A87 9C                             pushf
00002A88 FA                             cli
00002A89 53                             push   bx
                                        ;push  ax ; *           ; AL = cmos table address to be read
                                        ; 19/12/2023
00002A8A 89C3                           mov    bx, ax ; * ; input
00002A8C 0C80                           or     al, 80h
00002A8E E670                           out    70h, al          ; CMOS Memory/RTC Index      Register:
                                                                ; RTC Seconds
00002A90 90                             nop                     ; (undocumented delay needed)
00002A91 E471                           in     al, 71h          ; CMOS Memory/RTC Data Register

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

                                        ; 19/12/2023
                                        ; al = output, bl = input
00002A93 93                             xchg   ax, bx ; *
                                        ; bl = output, al = input

00002A94 2480                           and    al, 80h
00002A96 0C0F                           or     al, 0Fh
00002A98 E670                           out    70h, al          ; CMOS Memory/RTC Index      Register:
                                                                ; RTC Seconds
00002A9A 90                             nop
00002A9B E471                           in     al, 71h          ; CMOS Memory/RTC Data Register
                                        ;mov   ax, bx ; * ; output
                                        ; 19/12/2023
00002A9D 93                             xchg   ax, bx
00002A9E 5B                             pop    bx

                                        ; 19/12/2023
                                cmos_rw_popf:
00002A9F 0E                             push   cs               ; *place code segment in stack and
00002AA0 E80100                         call   cmos_popf        ; *handle popf for b- level 80286
00002AA3 C3                             retn                    ; return with flags restored

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

                                cmos_popf:
00002AA4 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

00002AA5 51                             push   cx ; 1*
00002AA6 57                             push   di ; 2*
                                        ;mov   [cs:doscnt], cx
00002AA7 890E[021A]                     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
00002AAB 89F5                           mov    bp, si

00002AAD 89D8                           mov    ax, bx

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

```
11163
11164                                   ;;sbb  [cs:ClusterH], 0
11165                                   ;sbb   [ClusterH], 0
11166 00002AB2 83DD00                   sbb    bp, 0
11167
11168                                   ; 20/12/2023
11169                                   ;;xchg ax, [cs:ClusterH]
11170                                   ;xchg  ax, [ClusterH]
11171 00002AB5 95                       xchg   ax, bp
11172
11173 00002AB6 F7E1                     mul    cx
11174
11175                                   ;;xchg ax, [cs:ClusterH]
11176                                   ;xchg  ax, [ClusterH]
11177 00002AB8 95                       xchg   ax, bp ; (+)
11178                                   ;
11179 00002AB9 F7E1                     mul    cx             ;; convert to logical sector
11180                                                          ;; dx:ax = matching logical sector number
11181                                                          ;;       starting from the data sector
11182                                   ;;add  ax, [cs:bios_l]
11183                                   ;;adc  dx, [cs:bios_h]       ; dx:ax = first logical sector to read
11184                                   ; 12/12/2023
11185                                   ;add   ax, [bios_l]
11186                                   ;adc   dx, [bios_h]   ; dx:ax = first logical sector to read
11187
11188                                   ; 20/12/2023
11189                                   ;;add  dx, [cs:ClusterH]
11190                                   ;add   ax, [cs:First_Data_Sector]
11191                                   ;adc   dx, [cs:First_Data_Sector+2]
11192 00002ABB 01EA                     add    dx, bp ; (+)
11193                                   ;add   dx, [ClusterH] ; convert to logical sector
11194                                                         ; dx:ax = matching logical sector number
11195                                                         ;       starting from the data sector
11196 00002ABD 0306[FE19]               add    ax, [First_Data_Sector]
11197 00002AC1 1316[001A]               adc    dx, [First_Data_Sector+2]
11198                                                         ; dx:ax = first logical sector to read
11199                                 unpack:
11200                                   ; 20/12/2023
11201 00002AC5 1E                       push   ds ; 3* ; ds = cs ; 12/12/2023
11202 00002AC6 52                       push   dx ; 4* ; * ; 12/12/2023
11203 00002AC7 50                       push   ax ; 5*
11204                                   ; 29/12/2023
11205 00002AC8 56                       push   si ; 6*
11206 00002AC9 53                       push   bx ; 7*
11207
11208                                   ;;mov  si, [cs:fatloc]
11209                                   ;mov   si, [fatloc] ; 12/12/2023
11210                                   ;mov   ds, si
11211                                   ; 20/12/2023
11212                                   ;mov   ax, [fatloc]
11213                                   ;mov   ds, ax
11214 00002ACA 53                       push   bx ; 8*
11215 00002ACB FF36[041A]               push   word [fatloc] ; 9*
11216
11217                                   ;test  byte [cs:fbigfat], 20h
11218 00002ACF F606[081A]20             test   byte [fbigfat], 20h   ; fbigbig FAT32 ?
11219 00002AD4 1F                       pop    ds ; 9* ; ds = [fatloc]
11220 00002AD5 7415                     jz      short not_32bit_cluster ; no
11221                                 unpack32:
11222                                   ;push  dx
11223 00002AD7 89F2                     mov    dx, si
11224                                   ;mov   si, bx
11225 00002AD9 5E                       pop    si ; 8* ; si = bx
11226 00002ADA 01F6                     add    si, si
11227 00002ADC 11D2                     adc    dx, dx
11228 00002ADE 01F6                     add    si, si
11229 00002AE0 11D2                     adc    dx, dx
11230                                                  ; dx:si = 4*(si:bx) ; clust num offset from FAT entry 0
11231 00002AE2 E89600                   call   get_fat_sector
11232 00002AE5 8B7702                   mov    si, [bx+2]     ; high word of the FAT32 cluster number
11233 00002AE8 8B1F                     mov    bx, [bx]       ; low word of the FAT32 cluster number
11234                                   ;pop   dx
11235 00002AEA EB45                     jmp    short getcl1
11236
11237                                 not_32bit_cluster:
11238                                   ;mov   si, bx         ; next cluster
11239 00002AEC 5E                       pop    si ; 8* ; si = bx
11240 00002AED 2EF606[081A]40           test   byte [cs:fbigfat], 40h; fbig
11241                                                  ; 16 bit fat?
11242 00002AF3 752F                     jnz    short unpack16 ; yes
11243                                 unpack12:
11244 00002AF5 D1EE                     shr    si, 1          ; 12 bit fat. si = si/2
11245                                                         ; si = clus + clus/2
11246 00002AF7 01DE                     add    si, bx         ;
11247                                                         ; (si = byte offset of the cluster in the FAT)
11248                                   ;push  dx ; 12/12/2023
11249 00002AF9 31D2                     xor    dx, dx
11250                                   ; 12/12/2023
11251                                   ; ds = FAT buffer segment
11252 00002AFB E87D00                   call   get_fat_sector
11253                                   ;pop   dx ; 12/12/2023
11254
11255 00002AFE 8B07                     mov    ax, [bx]       ; save it into ax
11256 00002B00 750C                     jnz    short even_odd ; if not a splitted fat, check even-odd.
11257                                   ; 25/06/2023
11258                                   ;mov   al, [bx]       ; splitted fat
11259
11260                                   ; 12/12/2023
11261                                   ;mov   [cs:temp_cluster], al
11262 00002B02 50                       push   ax ; **; al = low 8 bits of 12 bits cluster number
11263
11264 00002B03 46                       inc    si             ; (next byte)
11265
11266                                   ;push  dx ; 12/12/2023
11267 00002B04 31D2                     xor    dx, dx
11268 00002B06 E87200                   call   get_fat_sector
11269                                   ;pop   dx ; 12/12/2023
11270
11271                                   ;mov   al, ds:0
11272                                   ; 12/12/2023
11273                                   ; ds = FAT buffer segment
11274                                   ;mov   al, [0] ; 19/10/2022
11275                                   ;mov   [cs:temp_cluster+1], al
11276                                   ;mov   ax, [cs:temp_cluster]
11277                                   ; 12/12/2023
11278                                   ;mov   al, [cs:temp_cluster]
11279 00002B09 58                       pop    ax  ; ** ; al = low 8 bits of 12 bits cluster number
11280 00002B0A 8A260000                 mov    ah, [0] ; high 4 bits (bits 7 to 11) of 12 bits cluster num
11281                                 even_odd:
11282                                   ; 29/12/2023
11283 00002B0E 5B                       pop    bx ; 7*        ; restore old fat entry      value
11284 00002B0F 53                       push   bx             ; save it right       away.
11285 00002B10 D1EB                     shr    bx, 1          ; was it even or odd?
11286 00002B12 7308                     jnc    short havclus  ; it was even.
```

```
11287 00002B14 D1E8                             shr     ax, 1           ; odd. massage fat value and keep
11288                                                                   ; the highest 12 bits.
11289 00002B16 D1E8                             shr     ax, 1
11290 00002B18 D1E8                             shr     ax, 1
11291 00002B1A D1E8                             shr     ax, 1
11292                                   havclus:
11293 00002B1C 89C3                             mov     bx, ax          ; now bx = new fat entry.
11294 00002B1E 81E3FF0F                         and     bx, 0FFFh       ; keep low 12 bits.
11295 00002B22 EB0B                             jmp     short unpackx
11296                                   ; ---------------------------------------------------------------------------
11297
11298                                   unpack16:
11299                                           ;push   dx      ; 12/12/2023
11300 00002B24 31D2                             xor     dx, dx ; 0
11301 00002B26 D1E6                             shl     si, 1           ; extend to 32 bit offset
11302                                           ;adc    dx, 0
11303                                           ; 12/12/2023
11304 00002B28 D1D2                             rcl     dx, 1
11305
11306                                           ; 12/12/2023
11307                                           ; ds = FAT buffer segment
11308 00002B2A E84E00                           call    get_fat_sector
11309                                           ;pop    dx      ; 12/12/2023
11310 00002B2D 8B1F                             mov     bx, [bx]        ;
11311                                                                   ; bx = new fat entry.
11312                                   unpackx:
11313                                           ; 20/12/2023
11314 00002B2F 31F6                             xor     si, si          ; high word of cluster number = 0
11315                                                                   ; (FAT12 or FAT16)
11316                                   getcl1:
11317                                           ; 29/12/2023
11318 00002B31 58                               pop     ax      ; 7* - cluster number lw
11319                                           ;pop    word [cs:ClusterH]
11320 00002B32 5A                               pop     dx      ; 6* - cluster number hw
11321
11322                                           ; 20/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
11323                                           ; (this is a fast kernel loading method by the MSDOS programmer)
11324                                           ; ((consequtive clusters --> consequtive sectors))
11325
11326 00002B33 29D8                             sub     ax, bx  ; previous - current (or current - new)
11327                                           ;sbb    [cs:ClusterH], si
11328 00002B35 19F2                             sbb     dx, si
11329                                           ;;cmp   [cs:ClusterH], -1 ; one apart? (current = previous+1)
11330                                           ;cmp    dx, -1
11331                                           ; 29/12/2023
11332 00002B37 42                               inc     dx ; -1 -> 0
11333 00002B38 7501                             jnz     short not_consequental
11334                                           ;cmp    ax, -1          ; 0FFFFh ; is [ClusterH]:ax = -1 ?
11335 00002B3A 40                               inc     ax ; -1 -> 0
11336                                   not_consequental:
11337 00002B3B 58                               pop     ax ; 5*         ; restore logical sector (low)
11338 00002B3C 5A                               pop     dx ; 4* ; * ; 12/12/2023
11339 00002B3D 1F                               pop     ds ; 3*
11340
11341                                           ;; 12/12/2023
11342                                           ;; (this is a fast kernel loading method by the MSDOS programmer)
11343                                           ;; ((consequtive clusters --> consequtive sectors))
11344                                           ;; ds = cs
11345                                           ;sub    si, bx
11346                                           ;cmp    si, -1          ; one apart? (consequtive?)
11347                                           ;                       ; (current = previous+1)
11348
11349 00002B3E 7507                             jnz     short getcl2    ; no, read [doscnt] sectors
11350
11351                                           ;add    [cs:doscnt], cx ; (cx = sectors per cluster)
11352 00002B40 010E[021A]                       add     [doscnt], cx ; 12/12/2023 ; add to read count
11353 00002B44 E97EFF                           jmp     unpack
11354                                   ; ---------------------------------------------------------------------------
11355
11356                                   getcl2:
11357 00002B47 56                               push    si ; 20/12/2023
11358 00002B48 53                               push    bx
11359                                           ; bx = low word of the new cluster number
11360                                           ; 20/12/2023 - Retro DOS v5.0 (32 bit cluster numbers)
11361                                           ; si = high word of the new cluster number
11362 00002B49 52                               push    dx              ; sector to read (high word)
11363 00002B4A 50                               push    ax              ; sector to read (low word)
11364
11365                                           ; 12/12/2023
11366                                           ; ds = cs
11367                                           ;mov    ax, [cs:drvfat]       ; get drive and       fat spec
11368                                           ;mov    cx, [cs:doscnt]
11369 00002B4B A1[FC19]                         mov     ax, [drvfat]    ; get drive and       fat spec
11370
11371                                           ;;;
11372                                           ; 20/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
11373                                           ;
11374                                           ; dma and segment (64K boundary) overrun precaution
11375                                           ; (sector count will be decreased if it is required)
11376 00002B4E 89F9                             mov     cx, di
11377 00002B50 F7D1                             not     cx              ; cx = 65535 - cx
11378 00002B52 D1E9                             shr     cx, 1           ; cx = cx/2
11379 00002B54 30C9                             xor     cl, cl
11380 00002B56 86E9                             xchg    cl, ch          ; cx = cx/256
11381
11382                                           ;cmp    cx, [cs:doscnt]
11383                                                                   ; if sector read count > cx, decrease it to cx
11384 00002B58 3B0E[021A]                       cmp     cx, [doscnt]
11385 00002B5C 7604                             jbe     short getcl3
11386                                           ;;;
11387                                           ;mov    cx, [cs:doscnt]
11388 00002B5E 8B0E[021A]                       mov     cx, [doscnt]
11389                                   getcl3:
11390 00002B62 5A                               pop     dx              ; sector to read for diskrd (low)
11391                                           ;pop    word [cs:start_sec_h]
11392                                           ; 12/12/2023
11393 00002B63 8F06[9C04]                       pop     word [start_sec_h]
11394                                                                   ; sector to read for diskrd (high)
11395                                           ; 06/04/2024
11396                                           ;;;
11397 00002B67 51                               push    cx ; +*
11398                                           ;;;
11399
11400                                           ; 12/12/2023
11401                                           ; ds = cs
11402                                           ;push   ds
11403                                           ;push   cs
11404                                           ;pop    ds
11405
11406 00002B68 0E                               push    cs              ; simulate far call
11407
11408                                           ; 20/12/2023
11409                                           ; 17/10/2022
11410 00002B69 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

00002B6C E804EF                         call    call_bios_code ; read the clusters

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

                                        ; 06/04/2024
                                        ;;;
00002B6F 58                             pop     ax ; +*         ; sector count
                                        ;;;

00002B70 5B                             pop     bx              ; lw of the new cluster number
00002B71 5E                             pop     si ; 20/12/2023 ; hw of the new cluster number

00002B72 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]

00002B73 86C4                           xchg    ah, al          ; multiply by 256
00002B75 D1E0                           shl     ax, 1           ; times 2 equal 512
00002B77 01C7                           add     di, ax          ; update load location

00002B79 59                             pop     cx ; 1*         ; restore sectors/cluster

00002B7A C3                             retn

                                ; =============== S U B   R O U T I 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)
00002B7B 51                             push    cx ; read count (sectors per cluster)
00002B7C 57                             push    di ; IBMDOS.COM/MSDOS.SYS load offset
00002B7D 56                             push    si ; FAT offset value (from fat entry 0)
00002B7E 06                             push    es ; IBMDOS.COM/MSDOS.SYS load segment
00002B7F 1E                             push    ds ; FAT buffer segment

                                        ; 12/12/2023
00002B80 0E                             push    cs
00002B81 1F                             pop     ds

                                        ; 06/04/2024
                                        ; dx:si = offset value (starting from fat entry 0)
                                        ;             of fat entry to find

00002B82 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
00002B84 F736[0A1A]                     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 ; *
00002B88 50                             push    ax
                                        ;push   cs ; *
                                        ;pop    ds ; *

                                        ;mov    ax, [cs:drvfat]       ; get drive # and FAT id
                                        ; 12/12/2023
00002B89 A1[FC19]                       mov     ax, [drvfat]   ; get drive # and FAT id
00002B8C BD[A405]                       mov     bp, SETDRIVE
                                        ;mov    bp, 5AEh  ; PCDOS 7.1 IBMBIO.COM
                                        ;;mov   bp, 4D7h        ; setdrive
                                                                ; at 2C7h:4D7h = 70h:2A47h
00002B8F 0E                             push    cs              ; simulate far call
00002B90 E8E0EE                         call    call_bios_code ; get bds for drive
00002B93 58                             pop     ax              ; (sector number -without reserved and hidden sectors-)
00002B94 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
00002B98 3B06[0C1A]                     cmp     ax, [last_fat_sec_num]
                                        ;cmp    ax, [cs:last_fat_sec_num]
00002B9C 741C                           jz      short gfs_split_chk ; don't need to read it again.
00002B9E A3[0C1A]                       mov     [last_fat_sec_num], ax
                                        ;mov    [cs:last_fat_sec_num], ax
                                                                ; sector number
```

```
                                                        ; (in the partition, without hidden sectors)
                                                ; 13/12/2023
00002BA1 07                                     pop     es ; FAT buffer segment (DS on top of the stack)
00002BA2 06                                     push    es ; (put it on top of the stack again)

00002BA3 52                                     push    dx ; cluster number offset

                                                ; 12/12/2023
00002BA4 31C9                                   xor     cx, cx
00002BA6 890E[9C04]                             mov     [start_sec_h], cx ;0
                                                ;mov    word [cs:start_sec_h], 0
                                                        ; prepare to read the fat sector
                                                        ; start_sec_h is always       0 for fat sector.
00002BAA 89C2                                   mov     dx, ax
                                                ; 12/12/2023
00002BAC 41                                     inc     cx ; cx = 1
                                                ;mov    cx, 1           ; 1 sector read
                                                ;mov    ax, [cs:drvfat]
00002BAD A1[FC19]                               mov     ax, [drvfat]
                                                ;push   ds
                                                ;pop    es

00002BB0 31FF                                   xor     di, di  ; 0     ; es:di -> fatloc segment:0

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

00002BB2 0E                                     push    cs              ; simulate far call

                                                ; 20/12/2023
                                                ; 17/10/2022
00002BB3 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
00002BB6 E8BAEE                                 call    call_bios_code ; read the clusters

                                                ; 12/12/2023
                                                ;pop    ds
                                                ; ds = cs = biosdata segment

00002BB9 5A                                     pop     dx ; cluster number offset

                                        gfs_split_chk:
                                                ; 13/12/2023
                                                ;mov    cx, [cs:md_sectorsize] ; 512
00002BBA 8B0E[0A1A]                             mov     cx, [md_sectorsize]
                                        ;gfs_split_chk:
00002BBE 49                                     dec     cx              ; 511
00002BBF 39CA                                   cmp     dx, cx          ; if offset points to the
                                                                        ; last byte of this sector,
                                                                        ; then splitted       entry.
00002BC1 89D3                                   mov     bx, dx          ; set bx to dx

                                                ; 12/12/2023
                                                ; bx = dx = cluster number offset in the FAT buffer
00002BC3 1F                                     pop     ds ; FAT buffer segment
00002BC4 07                                     pop     es ; IBMDOS.COM/MSDOS.SYS load segment
00002BC5 5E                                     pop     si ; FAT offset value (from fat entry 0)
00002BC6 5F                                     pop     di ; IBMDOS.COM/MSDOS.SYS load offset
00002BC7 59                                     pop     cx ; read count (sectors per cluster)
                                                ;pop    ax

00002BC8 C3                                     retn

                                        %else

                                            ; 06/04/2024 - temporary
                                        temp_cluster:       dw      0

                                        ClusterH:   dw      0

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

                                        getclus:
                                                        push    cx              ; 1*
                                                                                ; si:bx = (32 bit) cluster to read
                                                                                ; cx = sectors per cluster
                                                                                ; es:di = load location
                                                        push    di              ; 2*
                                                        mov     [cs:doscnt], cx
                                                        mov     ax, bx
                                                        mov     [cs:ClusterH], si ; high word of cluster number
                                                        sub     ax, 2
                                                        sbb     word [cs:ClusterH], 0
                                                        xchg    ax, [cs:ClusterH]
                                                        mul     cx
                                                        xchg    ax, [cs:ClusterH]
                                                        mul     cx
                                                        add     dx, [cs:ClusterH] ; convert to logical sector
                                                                                ; dx:ax = matching logical sector number
                                                                                ; starting from the data sector
                                                        add     ax, [cs:First_Data_Sector]
                                                        adc     dx, [cs:First_Data_Sector+2]
                                                                                ; dx:ax = first logical sector to read
                                        unpack:
                                                        push    ds              ; 3*
                                                        push    ax              ; 4*
                                                        push    si              ; 5*
                                                        push    bx              ; 6*
                                                        mov     ax, [cs:fatloc]
                                                        mov     ds, ax
                                                        test    byte [cs:fbigfat], 20h ; fbigbig
                                                                                ; FAT32 ?
                                                        jz      short not_32bit_cluster ; no
                                        unpack32:                               ; yes
                                                        push    dx              ; 7*
                                                        mov     dx, si
                                                        mov     si, bx
                                                        add     si, si
                                                        adc     dx, dx
                                                        add     si, si
                                                        adc     dx, dx          ; dx:si = 4*(si:bx)
                                                        call    get_fat_sector
                                                        mov     si, [bx+2]      ; byte 16-31 of the FAT32 cluster number
                                                        mov     bx, [bx]        ; byte 0-15 of the FAT32 cluster number
                                                        pop     dx              ; 7*
                                                        jmp     short getcl1
                                        ; ---------------------------------------------------------------------------
```

```
11659                                   not_32bit_cluster:
11660                                                   mov     si, bx          ; next cluster
11661                                                   test    byte [cs:fbigfat], 40h ; fbig
11662                                                                           ; FAT16 ?
11663                                                   jnz     short unpack16  ; yes
11664                                   unpack12:
11665                                                   shr     si, 1
11666                                                   add     si, bx          ; 12 bit fat. si=si/2
11667                                                                           ; si = clus + clus/2
11668                                                                           ; (si = byte offset of the cluster in the FAT)
11669                                                   push    dx
11670                                                   xor     dx, dx
11671                                                   call    get_fat_sector
11672                                                   pop     dx
11673                                                   mov     ax, [bx]        ; save cluster number into ax
11674                                                   jnz     short even_odd  ; if not a splitted fat, check even-odd
11675                                                   mov     al, [bx]        ; (not needed!) Erdogan Tan - 2023
11676                                                   mov     byte [cs:temp_cluster], al ; splitted fat
11677                                                   inc     si              ; (next byte)
11678                                                   push    dx
11679                                                   xor     dx, dx
11680                                                   call    get_fat_sector
11681                                                   pop     dx
11682                                                   mov     al, [0]          ; mov ah,[0]
11683                                                   mov     byte [cs:temp_cluster+1], al
11684                                                   mov     ax, [cs:temp_cluster] ; mov al,[cs:temp_cluster]
11685                                   even_odd:
11686                                                   pop     bx              ; restore old fat entry value
11687                                                   push    bx              ; 6*
11688                                                   shr     bx, 1           ; was it even or odd?
11689                                                   jnb     short havclus   ; it was even
11690                                                   shr     ax, 1           ; odd. massage fat value and keep
11691                                                                           ; the highest 12 bits.
11692                                                   shr     ax, 1
11693                                                   shr     ax, 1
11694                                                   shr     ax, 1
11695                                   havclus:
11696                                                   mov     bx, ax          ; now bx = new fat entry
11697                                                   and     bx, 0FFFh       ; keep low 12 bits
11698                                                   jmp     short unpackx
11699                                   ; ---------------------------------------------------------------------------
11700
11701                                   unpack16:
11702                                                   push    dx
11703                                                   xor     dx, dx          ; extend to 32 bit offset
11704                                                   shl     si, 1           ; cluster number * 2
11705                                                   adc     dx, 0
11706                                                   call    get_fat_sector
11707                                                   pop     dx
11708                                                   mov     bx, [bx]        ; bx = new fat entry
11709                                   unpackx:
11710                                                   xor     si, si          ; high word of cluster number = 0
11711                                                                           ; (FAT12 or FAT16)
11712                                   getcl1:
11713                                                   pop     ax              ; 6* - cluster number lw
11714                                                   pop     word [cs:ClusterH] ; 5* - cluster number hw
11715                                                   sub     ax, bx          ; previous - current (or current - new)
11716                                                   sbb     [cs:ClusterH], si
11717                                                   cmp     word [cs:ClusterH], -1 ; one apart? (current = previous+1)
11718                                                   jnz     short not_consenquental
11719                                                   cmp     ax, -1          ; 0FFFFh ; is [ClusterH]:ax = -1 ?
11720                                   not_consenquental:
11721                                                   pop     ax              ; 4* - low word of first logical sector
11722                                                   pop     ds              ; 3*
11723                                                   jnz     short getcl2
11724                                                   add     [cs:doscnt], cx ; consequental cluster read, +1 cluster sectors
11725                                                                           ; (cx = sectors per cluster)
11726                                                   jmp     unpack
11727                                   ; ---------------------------------------------------------------------------
11728
11729                                   getcl2:
11730                                                   push    bx
11731                                                   push    si
11732                                                   push    dx              ; sector to read (high)
11733                                                   push    ax              ; sector to read (low)
11734                                                   mov     ax, [cs:drvfat] ; get drive and fat spec
11735                                                   mov     cx, di          ; dma and segment (64K boundary) overrun precaution
11736                                                                           ; (sector count will be decreased if it is required)
11737                                                   not     cx              ; cx = 65535 - cx
11738                                                   shr     cx, 1           ; cx = cx/2
11739                                                   xor     cl, cl
11740                                                   xchg    cl, ch          ; cx = cx/256
11741                                                   cmp     cx, [cs:doscnt] ; if sector read count > cx, decrease it to cx
11742                                                   jbe     short getcl3
11743                                                   mov     cx, [cs:doscnt]
11744                                   getcl3:
11745                                                   pop     dx              ; sector to read for diskrd (low)
11746                                                   pop     word [cs:start_sec_h] ; sector to read for diskrd (high)
11747                                                   push    cx
11748                                                   push    ds
11749                                                   push    cs
11750                                                   pop     ds
11751                                                   push    cs              ; simulate far call
11752                                                   mov     bp, DISKRD ; BIOSCODE:0A2Bh ; 364h:0A2Bh
11753                                                   call    call_bios_code
11754                                                   pop     ds
11755                                                   pop     ax              ; sector count
11756                                                   pop     si
11757                                                   pop     bx
11758                                                   pop     di              ; 2* - load location (es:di)
11759                                                   xchg    ah, al
11760                                                   shl     ax, 1           ; ax = ax * 512 ; byte count
11761                                                   add     di, ax          ; update load location
11762                                                   pop     cx              ; 1* - restore sectors/cluster
11763                                                   retn
11764
11765                                   ; =============== S U B R O U T I N E =======================================
11766
11767                                   get_fat_sector:
11768                                                   push    ax              ; dx:si = offset value (starting from fat entry 0)
11769                                                                           ;         of fat entry to find
11770                                                   push    cx
11771                                                   push    di
11772                                                   push    si
11773                                                   push    es
11774                                                   push    ds
11775                                                   mov     ax, si
11776                                                   mov     cx, [cs:md_sectorsize] ; 512
11777                                                   div     cx
11778                                                   nop                     ; ax = sector number, dx = offset
11779                                                   push    es
11780                                                   push    ds
11781                                                   push    di
11782                                                   push    ax
```

```
11783                                           push    cs
11784                                           pop     ds
11785                                           mov     ax, [cs:drvfat] ; get drive # and FAT id
11786                                           mov     bp, SetDrive ; BIOSCODE:05AEh
11787                                           push    cs              ; simulate far call
11788                                           call    call_bios_code  ; get bds for drive
11789                                           pop     ax              ; (sector number -without reserved and hidden sectors-)
11790                                           add     ax, [es:di+9]   ; [es:di+BDS.resectors]
11791                                                                   ; add #reserved_sectors
11792                                           pop     di
11793                                           pop     ds
11794                                           pop     es
11795                                           cmp     ax, [cs:last_fat_sec_num]
11796                                           jz      short gfs_split_chk ; don't need to read it again
11797                                           mov     [cs:last_fat_sec_num], ax ; sector number
11798                                                                   ; (in the partition, without hidden sectors)
11799                                           push    dx
11800                                           mov     word [cs:start_sec_h], 0 ; prepare to read the fat sector
11801                                                                   ; start_sec_h is always 0 for fat sector
11802                                           mov     dx, ax
11803                                           mov     cx, 1           ; 1 sector read
11804                                           mov     ax, [cs:drvfat]
11805                                           push    ds
11806                                           pop     es
11807                                           xor     di, di          ; es:di -> fatloc segment:0
11808                                           push    ds
11809                                           push    cs
11810                                           pop     ds
11811                                           push    cs              ; simulate far call
11812                                           mov     bp, DISKRD ; BIOSCODE:0A2Bh ; 364h:0A2Bh
11813                                           call    call_bios_code
11814                                           pop     ds
11815                                           pop     dx
11816                                           mov     cx, [cs:md_sectorsize] ; 512
11817                                   gfs_split_chk:
11818                                           dec     cx              ; 511
11819                                           cmp     dx, cx          ; if offset points to the last byte of this sector,
11820                                                                   ; then splitted entry.
11821                                           mov     bx, dx          ; offset value from fatloc segment
11822                                           pop     ds
11823                                           pop     es
11824                                           pop     si
11825                                           pop     di
11826                                           pop     cx
11827                                           pop     ax
11828                                           retn
11829
11830                                   %endif
11831
11832                                   ; 15/10/2022
11833                                   ;Bios_Data_Init    ends
11834
11835                                   ; ---------------------------------------------------------------------------
11836
11837                                               ; 09/12/2022
11838                                               ;db 0
11839
11840                                   numbertodiv equ ($-BData_start)
11841                                   numbertomod equ (numbertodiv % 16)
11842
11843                                   %if numbertomod>0 & numbertomod<16
11844 00002BC9 00<rep 7h>                             times (16-numbertomod) db 0
11845                                   %endif
11846
11847                                   ;align 16
11848
11849                                   ; 09/12/2022
11850                                   IOSYSCODESEGOFF    equ $ - BData_start
11851                                   IOSYSCODESEG       equ (IOSYSCODESEGOFF>>4)+(700h>>4)
11852
11853                                   ;--- End of DOSBIOS data segment ----------------------------------------
11854                                   ; ---------------------------------------------------------------------------
11855                                               ;db 4 dup(0)
11856                                   ; 09/12/2022
11857                                   ;           times 4 db 0   ; 19/10/2022
11858                                   ; ---------------------------------------------------------------------------
11859
11860                                   ;============================================================================
11861                                   ; DOS BIOS (IO.SYS) CODE SEGMENT
11862                                   ;============================================================================
11863                                   ; 09/12/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
11864                                   ; 30/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
11865
11866                                   section .BIOSCODE vstart=0
11867
11868                                   ; 30/12/2022
11869                                   ; (BIOSCODE SEGMENT is 2CCh for MSDOS 6.21 IO.SYS)  -- ((25C0h+700h)>>4) --
11870
11871                                   BCode_start:        ; 09/12/2022
11872
11873                                   ; 02/10/2022
11874
11875                                   ;--- DOSBIOS code segment -----------------------------------------------
11876                                   ;---------------------------------------------------------------------------
11877                                   ; MSBIO1.ASM (MSDOS 6.0, 1991)
11878                                   ;---------------------------------------------------------------------------
11879
11880                                   DOSBIOSEG_2C7h:    ;db 30h dup(0)        ; SEGMENT 2C7h (2C70h-700h=2570h)
11881 00000000 00<rep 30h>                            times 48 db 0          ; 19/10/2022
11882 00000030 7000                    BiosDataWord:      dw 70h
11883
11884                                   ; 15/10/2022
11885                                   ;BIOSDATAWORD      equ BiosDataWord - DOSBIOSEG_2C7h
11886                                   ; 09/12/2022
11887                                   BIOSDATAWORD       equ BiosDataWord
11888
11889                                   ; ---------------------------------------------------------------------------
11890
11891                                   ; 15/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
11892                                   ; 20/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
11893
11894                                   ;****************************************************************************
11895                                   ;*                                                                          *
11896                                   ;*  seg_reinit is called with ax = our new code segment value,              *
11897                                   ;*    trashes di, cx, es                                                    *
11898                                   ;*                                                                          *
11899                                   ;*  cas -- should be made disposable!                                       *
11900                                   ;*                                                                          *
11901                                   ;****************************************************************************
11902
11903                                       ; 20/09/2023
11904                                       ; 10/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
11905                                       ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0032h
11906
```

```
                                _seg_reinit:
00000032 2E8E06[3000]                   mov     es, [cs:BIOSDATAWORD]
                                                        ; at 2C7h:30h or 70h:25A0h
                                        ;mov    di, (offset cdev+2)
00000037 BF[0406]                       mov     di, cdev+2      ; 19/10/2022
                                        ;mov    cx, 4           ; (end_BC_entries - cdev)/4
                                        ; 10/08/2023
0000003A B90300                         mov     cx, 3 ; (PCDOS 7.1)
                                _seg_reinit_1:
0000003D AB                             stosw                   ; modify Bios_Code entry points
0000003E 47                             inc     di
0000003F 47                             inc     di
00000040 E2FB                           loop    _seg_reinit_1
                                        ; 10/08/2023 (PCDOS 7.1)
                                        ; (direct jump to i2f_handler from BIOSDATA:bios_i2f)
                                        ; (instead of 'bcode_i2f: dw i2f_handler, IOSYSCODESEG')
00000042 26A3[0800]                     mov     [es:bios_i2f_seg], ax ; actual BIOSCODE segment

00000046 CB                             retf

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

                                ; 15/10/2022

                                ;****************************************************************************
                                ;*                                                                          *
                                ;*  chardev_entry - main device driver dispatch routine                     *
                                ;*     called with a dummy parameter block on the stack                     *
                                ;*     dw dispatch_table, dw prn/aux numbers (optional)                     *
                                ;*                                                                          *
                                ;*  will eventually take care of doing the transitions in                       *
                                ;*     out of Bios_Code                                                     *
                                ;*                                                                          *
                                ;****************************************************************************

                                        ; 20/09/2023
                                chardev_entry:                          ; 0070h:25B3h =       02C7h:0043h
00000047 56                             push    si
00000048 50                             push    ax
00000049 51                             push    cx
0000004A 52                             push    dx
0000004B 57                             push    di
0000004C 55                             push    bp
0000004D 1E                             push    ds
0000004E 06                             push    es
0000004F 53                             push    bx
00000050 89E5                           mov     bp, sp
00000052 8B7612                         mov     si, [bp+18]     ; get return address (dispatch table)
                                        ;;mov   ds, word [cs:0030h]
                                        ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:0030h]
00000055 2E8E1E[3000]                   mov     ds, [cs:BIOSDATAWORD] ; 17/10/2022
                                        ; 20/09/2023 (PCDOS 7.1 - IBMBIO.COM - BIOSCODE:005Ah)
0000005A C434                           les     si, [si]
                                        ;mov    ax, [si+2]      ; get the device number       if present
0000005C 8CC0                           mov     ax, es
0000005E A2[2100]                       mov     [auxnum], al
00000061 8826[8004]                     mov     [printdev], ah
                                        ;mov    si, [si]        ; point to the device dispatch table
00000065 C41E[1200]                     les     bx, [ptrsav]    ; get pointer to i/o packet
00000069 268A4701                       mov     al, [es:bx+1]  ; [es:bx+unit] ; al = unit code
0000006D 268A670D                       mov     ah, [es:bx+13] ; [es:bx+media]        ; ah = media descrip
00000071 268B4F12                       mov     cx, [es:bx+18] ; [es:bx+count]        ; cx = count
00000075 268B5714                       mov     dx, [es:bx+20] ; [es:bx+start]        ; dx = start sector
                                        ; 17/10/2022
00000079 81FE[6F05]                     cmp     si, DSKTBL
                                        ;;cmp   si, 579h        ; (PCDOS 7.1 IBMBIO.COM)
                                        ;cmp    si, 4A2h        ; dsktbl
                                                                ; at 2C7h:4A2h = 70h:2A12h
0000007D 7517                           jnz     short no_sector32_mapping

                                ; Special case for 32-bit start sector number:
                                ;   if (si==dsktbl) /* if this is a disk device call */
                                ;      set high 16 bits of secnum to 0
                                ;      if (secnum == 0xffff) fetch 32 bit sector number
                                ;
                                ; pass high word of sector number in start_sec_h, low word in dx
                                ;
                                ; note: start_l and start_h are the offsets within the io_request packet
                                ;     which contain the low and hi words of the 32 bit start sector if
                                ;     it has been used.
                                ;
                                ; note: remember not to destroy the registers which have been set up before

                                        ; 20/09/2023
                                        ;mov    ds:start_sec_h,         0 ; initialize to 0
0000007F C706[9C04]0000                 mov     word [start_sec_h], 0
00000085 83FAFF                         cmp     dx, 0FFFFh
00000088 750C                           jnz     short no_sector32_mapping
0000008A 268B571C                       mov     dx, [es:bx+28] ; [es:bx+start_h]
                                                                ; 32 bits dsk req
                                        ;mov    ds:start_sec_h,         dx ; start_sec_h = packet.start_h
0000008E 8916[9C04]                     mov     [start_sec_h], dx
00000092 268B571A                       mov     dx, [es:bx+26] ; [es:bx+start_l]
                                                                ; dx = packet.start_l
                                no_sector32_mapping:
00000096 97                             xchg    ax, di
00000097 268A4702                       mov     al, [es:bx+2]  ; [es:bx+cmd]
0000009B 2E3A04                         cmp     al, [cs:si]
0000009E 732B                           jnb     short command_error
000000A0 98                             cbw                     ; note that al <= 15 means ok
000000A1 D1E0                           shl     ax, 1
000000A3 01C6                           add     si, ax
000000A5 97                             xchg    ax, di
000000A6 26C47F0E                       les     di, [es:bx+14] ; [es:bx+trans]
000000AA FC                             cld
                                        ; 17/10/2022
000000AB 2EFF5401                       call    near [cs:si+1]
                                        ;call   word ptr cs:si+1
000000AF 7202                           jb      short already_got_ah_status
000000B1 B401                           mov     ah, 1
                                already_got_ah_status:
                                        ;;mov   ds, [cs:0030h] ; 15/10/2022
                                        ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:0030h]
                                                                ; cas note: shouldn't be needed!
000000B3 2E8E1E[3000]                   mov     ds, [cs:BIOSDATAWORD] ; 17/10/2022
                                        ;lds    bx, ds:ptrsav
000000B8 C51E[1200]                     lds     bx, [ptrsav]
000000BC 894703                         mov     [bx+3], ax      ; [bx+status]
                                                                ; mark operation complete
000000BF 5B                             pop     bx
000000C0 07                             pop     es
000000C1 1F                             pop     ds
000000C2 5D                             pop     bp
000000C3 5F                             pop     di
```

```
                                        pop     dx
                                        pop     cx
                                        pop     ax
                                        pop     si
                                        ;add    sp, 2           ; get rid of fake return address
                                        ; 20/09/2023 (PCDOS 7.1       - IBMBIO.COM - BIOSCODE:00C8h)
000000C8 44                             inc     sp
000000C9 44                             inc     sp

                                        ; fall through into bc_retf
                                ; ---------------------------------------------------------------------------
                                bc_retf:
000000CA CB                             retf
                                ; ---------------------------------------------------------------------------

                                command_error:
000000CB E80700                         call    bc_cmderr
000000CE EBE3                           jmp     short already_got_ah_status
                                ; 15/10/2022
                                ; 01/05/2019

                                ;---------------------------------------------------------------------------
                                ; The following piece of hack is for supporting CP/M compatibility
                                ; Basically at offset 5 we have a far call into 0:c0. But this does not call
                                ; 0:c0 directly instead it call f01d:fef0, because it needs to support 'lhld 6'
                                ; The following hack has to reside at ffff:d0 (= f01d:fef0) if BIOS is loaded
                                ; high.
                                ;---------------------------------------------------------------------------

                                        ;db 7 dup(0)

                                        ; 20/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
                                        ; (PCDOS 7.1 - IBMBIO.COM - BIOSCODE:0D0h)
                                        ; 15/10/2022
                                        ;dw 0           ; pad to bring offset to 0D0h

000000D0 00<rep 5h>             off_d0:     times 5 db 0   ; 5 bytes from 0:c0 will be copied onto here
                                                           ;  which is the CP/M call 5 entry point

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

                                ;   exit - all routines return through this path

                                bc_cmderr:
000000D5 B003                           mov     al, 3           ; 2C7h:D5h = 70h:2645h
                                                                ; unknown command error

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

                                ;   now zero the count field by subtracting its current value,
                                ;     which is still in cx, from itself.

                                ;   subtract the number of i/o's NOT YET COMPLETED from total
                                ;     in order to return the number actually complete

                                bc_err_cnt:
                                        ;les    bx, ds:ptrsav
                                        ; 19/10/2022
000000D7 C41E[1200]                     les     bx, [ptrsav]
000000DB 26294F12                       sub     [es:bx+18], cx ; [es:bx+count]
                                                               ; # of successful i/o's
000000DF B481                           mov     ah, 81h        ; mark error return
000000E1 F9                             stc                    ; indicate abnormal end
000000E2 C3                             retn

                                ; 15/10/2022

                                ;Bios_Code ends

                                ;---------------------------------------------------------------------------
                                ; MSCHAR.ASM - MSDOS 6.0 - 1991
                                ;---------------------------------------------------------------------------
                                ; 15/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ; 10/01/2019 - Retro DOS v4.0

                                ; 30/04/2019

                                ;title      mschar - character and clock devices

                                ;MODE_CTRLBRK       equ     0FFh

                                ; BIOSCODE:00E4h (MSDOS 6.21, IO.SYS)

                                ;****************************************************************************
                                ;*                                                                          *
                                ;*  device driver dispatch tables                                           *
                                ;*                                                                          *
                                ;*  each table starts with a byte which lists the number of                 *
                                ;*  legal functions, followed by that number of words. Each                 *
                                ;*  word represents an offset of a routine in Bios_Code which  *
                                ;*  handles the function. The functions are terminated with                 *
                                ;*  a near return. If carry is reset, a 'done' code is returned             *
                                ;*  to the caller. If carry is set, the ah/al registers are                 *
                                ;*  returned as abnormal completion status. Notice that ds                  *
                                ;*  is assumed to point to the Bios_Data segment throughout.   *
                                ;*                                                                          *
                                ;****************************************************************************

                                        ; 20/09/2023
                                        ; (PCDOS 7.1 - IBMBIO.COM - BIOSCODE:00E3h)
                                        ; 13/12/2022
000000E3 00                             db 0

                                        ; 13/12/2022
000000E4 0B                     con_table:  db ((con_table_end - con_table)-1)/2 ; 11
                                                                ; 2C7h:0E4h = 70h:2654h
000000E5 [FA01]                         dw bc_exvec  ; 1FBh     ; bc_exvec at 2C7h:1FBh       = 70h:276Bh
                                                                ; 00 init
000000E7 [FA01]                         dw bc_exvec  ; 1FBh     ; 01
000000E9 [FA01]                         dw bc_exvec  ; 1FBh     ; 02
000000EB [D500]                         dw bc_cmderr ; 0D5h     ; bc_exvec at 2C7h:D5h = 70h:2645h
                                                                ; 03
000000ED [5C01]                         dw con_read  ; 15Ch     ; con_read at 2C7h:15Ch       = 70h:26CCh
                                                                ; 04
000000EF [9F01]                         dw con_rdnd  ; 19Fh     ; con_rdnd at 2C7h:19Fh       = 70h:270Fh
                                                                ; 05
000000F1 [FA01]                         dw bc_exvec  ; 1FBh     ; 06
000000F3 [0802]                         dw con_flush ; 209h     ; con_flush at 2C7h:209h = 70h:2779h
                                                                ; 07
000000F5 [FC01]                         dw con_writ  ; 1FDh     ; con_writ at 2C7h:1FDh       = 70h:276Dh
                                                                ; 08
000000F7 [FC01]                         dw con_writ  ; 1FDh     ; 09
000000F9 [FA01]                         dw bc_exvec  ; 1FBh     ; 0A
```

```
12155                                   con_table_end:
12156 000000FB 1A                       prn_table:  db ((prn_table_end - prn_table)-1)/2 ; 26
12157                                               ; 2C7h:0FBh = 70h:266Bh
12158 000000FC [FA01]                               dw bc_exvec   ; 1FBh  ; bc_exvec
12159 000000FE [FA01]                               dw bc_exvec   ; 1FBh  ; 01
12160 00000100 [FA01]                               dw bc_exvec   ; 1FBh  ; 02
12161 00000102 [D500]                               dw bc_cmderr  ;       0D5h    ; bc_cmderr
12162 00000104 [1902]                               dw prn_input  ;       21Ah    ; prn_input
12163                                               ; 04 indicate zero chars read
12164 00000106 [C701]                               dw z_bus_exit ; 1C8h  ; z_bus_exit
12165                                               ; 05 read non-destructive
12166 00000108 [FA01]                               dw bc_exvec   ; 1FBh  ; 06
12167 0000010A [FA01]                               dw bc_exvec   ; 1FBh  ; 07
12168 0000010C [1E02]                               dw prn_writ   ;       21Fh    ; prn_writ
12169 0000010E [1E02]                               dw prn_writ   ; 21Fh  ; 09
12170 00000110 [4F02]                               dw prn_stat   ; 251h  ; prn_stat
12171 00000112 [FA01]                               dw bc_exvec   ; 1FBh  ; 0B
12172 00000114 [FA01]                               dw bc_exvec   ; 1FBh  ; 0C
12173 00000116 [FA01]                               dw bc_exvec   ; 1FBh  ; 0D
12174 00000118 [FA01]                               dw bc_exvec   ; 1FBh  ; 0E
12175 0000011A [FA01]                               dw bc_exvec   ; 1FBh  ; 0F
12176 0000011C [9402]                               dw prn_tilbusy ; 28Bh ; prn_tilbusy
12177 0000011E [FA01]                               dw bc_exvec   ; 1FBh  ; 11
12178 00000120 [FA01]                               dw bc_exvec   ; 1FBh  ; 12
12179 00000122 [C202]                               dw prn_genioctl ; 2BAh; prn_genioctl
12180 00000124 [FA01]                               dw bc_exvec   ; 1FBh  ; 14
12181 00000126 [FA01]                               dw bc_exvec   ; 1FBh  ; 15
12182 00000128 [FA01]                               dw bc_exvec   ; 1FBh  ; 16
12183 0000012A [FA01]                               dw bc_exvec   ; 1FBh  ; 17
12184 0000012C [FA01]                               dw bc_exvec   ; 1FBh  ; 18
12185 0000012E [F702]                               dw prn_ioctl_query ; 2F0h ; prn_ioctl_query
12186                                   prn_table_end:
12187 00000130 0B                       aux_table:  db ((aux_table_end - aux_table)-1)/2 ; 11
12188                                               ; 2C7h:130h = 70h:26A0h
12189 00000131 [FA01]                               dw bc_exvec   ; 1FBh  ; 00 - init
12190 00000133 [FA01]                               dw bc_exvec   ; 1FBh  ; 01
12191 00000135 [FA01]                               dw bc_exvec   ; 1FBh  ; 02
12192 00000137 [D500]                               dw bc_cmderr  ;       0D5h    ; 03
12193 00000139 [1203]                               dw aux_read   ; 30Dh  ; aux_read ; 04       - read
12194 0000013B [3703]                               dw aux_rdnd   ; 335h  ; aux_rdnd - 05       - read non-destructive
12195 0000013D [FA01]                               dw bc_exvec   ; 1FBh  ; 06
12196 0000013F [7803]                               dw aux_flsh   ;       36Ch    ; aux_flsh
12197 00000141 [7F03]                               dw aux_writ   ;       374h    ; aux_writ
12198 00000143 [7F03]                               dw aux_writ   ;       374h    ; 09
12199 00000145 [5703]                               dw aux_wrst   ;       355h    ; aux_wrst
12200                                   aux_table_end:
12201 00000147 0A                       tim_table   db ((tim_table_end - tim_table)-1)/2 ; 10
12202                                               ; 2C7h:147h = 70h:26B7h
12203 00000148 [FA01]                               dw bc_exvec   ; 1FBh  ; 00
12204 0000014A [FA01]                               dw bc_exvec   ; 1FBh  ; 01
12205 0000014C [FA01]                               dw bc_exvec   ; 1FBh  ; 02
12206 0000014E [D500]                               dw bc_cmderr  ;       0D5h    ; 03
12207 00000150 [E404]                               dw tim_read   ;       435h    ; tim_read
12208 00000152 [C701]                               dw z_bus_exit ; 1C8h  ; z_bus_exit
12209 00000154 [FA01]                               dw bc_exvec   ; 1FBh  ; 06
12210 00000156 [FA01]                               dw bc_exvec   ; 1FBh  ; 07
12211 00000158 [E503]                               dw tim_writ   ; 3DBh  ; tim_writ
12212 0000015A [E503]                               dw tim_writ   ; 3DBh  ; 09
12213                                   tim_table_end:
12214
12215                                   ; ---------------------------------------------------------------------------
12216
12217                                   ;****************************************************************************
12218                                   ;*                                                                          *
12219                                   ;*  con_read - read cx bytes from keyboard into buffer at es:di             *
12220                                   ;*                                                                          *
12221                                   ;****************************************************************************
12222
12223                                   con_read:                     ; 2C7h:15Ch = 70h:26CCh
12224                                               ;jcxz   short con_exit ; read cx bytes       from keyboard into buffer
12225 0000015C E306                               jcxz    con_exit      ; 19/10/2022
12226                                   con_loop:
12227 0000015E E80500                             call    chrin         ; get char in al
12228 00000161 AA                                 stosb                 ; store char at es:di
12229 00000162 E2FA                               loop    con_loop
12230                                   con_exit:
12231 00000164 F8                                 clc
12232 00000165 C3                                 retn
12233
12234                                   ; =============== S U B   R O U T I N E =======================================
12235
12236                                   ;****************************************************************************
12237                                   ;*                                                                          *
12238                                   ;*  chrin - input single char from keyboard into al                         *
12239                                   ;*                                                                          *
12240                                   ;*    we are going to issue extended keyboard function, if                  *
12241                                   ;*    supported. the returning value of the extended keystroke *
12242                                   ;*    of the extended keyboard function uses 0E0h in al                     *
12243                                   ;*    instead of 00h as in the conventional keyboard function. *
12244                                   ;*    this creates a conflict when the user entered real                    *
12245                                   ;*    greek alpha charater (= 0E0h) to  distinguish the extended            *
12246                                   ;*    keystroke and the greek alpha. this case will be handled *
12247                                   ;*    in the following manner:                                              *
12248                                   ;*                                                                          *
12249                                   ;*        ah = 16h                                                          *
12250                                   ;*        int 16h                                                           *
12251                                   ;*        if al == 0, then extended code (in ah)                            *
12252                                   ;*        else if al == 0E0h, then                                          *
12253                                   ;*        if ah <> 0, then extended code (in ah)                            *
12254                                   ;*         else greek_alpha character.                                      *
12255                                   ;*                                                                          *
12256                                   ;*  also, for compatibility reason, if an extended code is                  *
12257                                   ;*    detected, then we are going to change the value in al                 *
12258                                   ;*    from 0E0h to 00h.                                                     *
12259                                   ;*                                                                          *
12260                                   ;****************************************************************************
12261
12262                                               ; 19/10/2022
12263                                   chrin:
12264 00000166 8A26[7E04]                         mov     ah, [keyrd_func] ; set by msinit. 0 or 10h
12265 0000016A 30C0                               xor     al, al
12266 0000016C 8606[0C00]                         xchg    al, [altah]    ; get character       & zero altah
12267 00000170 08C0                               or      al, al
12268 00000172 752A                               jnz     short keyret
12269 00000174 CD16                               int     16h            ; KEYBOARD -
12270 00000176 09C0                               or      ax, ax
12271 00000178 74EC                               jz      short chrin
12272 0000017A 3D0072                             cmp     ax, 7200h      ; check for ctrl-prtsc
12273 0000017D 7504                               jnz     short alt_ext_chk
12274 0000017F B010                               mov     al, 10h
12275 00000181 EB1B                               jmp     short keyret
12276                                   ; ---------------------------------------------------------------------------
12277
12278                                   ;  if operation was extended function (i.e. keyrd_func != 0) then
```

```
12279                                   ;    if character read was 0E0h then
12280                                   ;      if extended byte was zero (i.e. ah == 0) then
12281                                   ;    goto keyret
12282                                   ;      else
12283                                   ;    set al to zero
12284                                   ;    goto alt_save
12285                                   ;      endif
12286                                   ;    endif
12287                                   ;  endif
12288                                   
12289                                   alt_ext_chk:
12290 00000183 803E[7E04]00                       cmp     byte [keyrd_func], 0
12291 00000188 740C                               jz      short not_ext
12292 0000018A 3CE0                               cmp     al, 0E0h
12293 0000018C 7508                               jnz     short not_ext
12294 0000018E 08E4                               or      ah, ah
12295 00000190 740C                               jz      short keyret
12296 00000192 30C0                               xor     al, al
12297 00000194 EB04                               jmp     short alt_save
12298                                   ; ---------------------------------------------------------------------------
12299                                   
12300                                   not_ext:
12301 00000196 08C0                               or      al, al          ; special case?
12302 00000198 7504                               jnz     short keyret
12303                                   alt_save:
12304 0000019A 8826[0C00]                         mov     [altah], ah     ; store special key
12305                                   keyret:
12306 0000019E C3                                 retn
12307                                   
12308                                   ; ---------------------------------------------------------------------------
12309                                   
12310                                   ;****************************************************************************
12311                                   ;*                                                                          *
12312                                   ;*  con_rdnd - keyboard non destructive read, no wait                       *
12313                                   ;*                                                                          *
12314                                   ;*  pc-convertible-type machine: if bit 10 is set by the dos                *
12315                                   ;*  in the status word of the request packet, and there is no               *
12316                                   ;*  character in the input buffer, the driver issues a system               *
12317                                   ;*  wait request to the rom. on return from the rom, it returns             *
12318                                   ;*  a 'char-not-found' to the dos.                                          *
12319                                   ;*                                                                          *
12320                                   ;****************************************************************************
12321                                   
12322                                             ; 19/10/2022
12323                                   con_rdnd:
12324 0000019F A0[0C00]                           mov     al, [altah]
12325 000001A2 08C0                               or      al, al
12326 000001A4 754C                               jnz     short rdexit
12327 000001A6 8A26[7F04]                         mov     ah, [keysts_func]
12328 000001AA CD16                               int     16h             ; KEYBOARD -
12329 000001AC 751D                               jnz     short gotchr
12330 000001AE 803E[7900]00                       cmp     byte [fhavek09], 0
12331 000001B3 7412                               jz      short z_bus_exit
12332 000001B5 C41E[1200]                         les     bx, [ptrsav]
12333                                             ; 12/12/2022
12334 000001B9 26F6470404                         test    byte [es:bx+4], 04h
12335                                             ;test   word [es:bx+3], 400h ; [es:bx+status]
12336 000001BE 7407                               jz      short z_bus_exit
12337 000001C0 B80041                             mov     ax, 4100h
12338 000001C3 30DB                               xor     bl, bl
12339 000001C5 CD15                               int     15h             ; SYSTEM - WAIT       ON EXTERNAL EVENT (CONVERTIBLE)
12340                                                                     ; AL = condition type, BH = condition compare or mask value
12341                                                                     ; BL = timeout value times 55 milliseconds, 00h     means no timeout
12342                                                                     ; DX = I/O port       address if AL bit 4 set
12343                                   z_bus_exit:
12344 000001C7 F9                                 stc                     ; 2C7h:1C8h = 70h:2738h
12345 000001C8 B403                               mov     ah, 3           ; indicate busy       status
12346 000001CA C3                                 retn
12347                                   ; ---------------------------------------------------------------------------
12348                                   
12349                                   gotchr:
12350 000001CB 09C0                               or      ax, ax
12351 000001CD 7508                               jnz     short notbrk    ; check for null after break
12352 000001CF 8A26[7E04]                         mov     ah, [keyrd_func] ; issue keyboard read function
12353 000001D3 CD16                               int     16h             ; KEYBOARD -
12354 000001D5 EBC8                               jmp     short con_rdnd ; get a real status
12355                                   ; ---------------------------------------------------------------------------
12356                                   
12357                                   notbrk:
12358 000001D7 3D0072                             cmp     ax, 7200h       ; check for ctrl-prtsc
12359 000001DA 7504                               jnz     short rd_ext_chk
12360 000001DC B010                               mov     al, 10h         ; ('P' & 1Fh) ; return control p
12361 000001DE EB12                               jmp     short rdexit
12362                                   ; ---------------------------------------------------------------------------
12363                                   
12364                                   rd_ext_chk:
12365 000001E0 803E[7E04]00                       cmp     byte [keyrd_func], 0 ; extended keyboard function?
12366 000001E5 740B                               jz      short rdexit
12367 000001E7 3CE0                               cmp     al, 0E0h        ; extended key value or       greek alpha?
12368 000001E9 7507                               jnz     short rdexit
12369 000001EB 80FC00                             cmp     ah, 0           ; scan code exist?
12370 000001EE 7402                               jz      short rdexit    ; yes. greek alpha char.
12371 000001F0 B000                               mov     al, 0           ; no. extended key stroke.
12372                                                                     ; change it for       compatibility
12373                                   rdexit:
12374 000001F2 C41E[1200]                         les     bx, [ptrsav]
12375 000001F6 2688470D                           mov     [es:bx+13], al ; [es:bx+media]
12376                                                                     ; return keyboard character here
12377                                   bc_exvec:
12378 000001FA F8                                 clc                     ; bc_exvec at 2C7h:1FBh       = 70h:276Bh
12379                                                                     ; indicate normal termination
12380 000001FB C3                                 retn
12381                                   ; ---------------------------------------------------------------------------
12382                                   
12383                                   ;****************************************************************************
12384                                   ;*                                                                          *
12385                                   ;*  con_write - console write routine                                       *
12386                                   ;*                                                                          *
12387                                   ;*  entry:  es:di -> buffer                                                         *
12388                                   ;*          cx    =  count                                                  *
12389                                   ;*                                                                          *
12390                                   ;****************************************************************************
12391                                   
12392                                   con_writ:
12393                                             ;jcxz   short bc_exvec
12394 000001FC E3FC                               jcxz    bc_exvec        ; 19/10/2022
12395                                             ; 12/12/2022
12396                                             ;jcxz   cc_ret
12397                                   con_lp:
12398 000001FE 268A05                             mov     al, [es:di]
12399 00000201 47                                 inc     di
12400 00000202 CD29                               int     29h             ; DOS 2+ internal - FAST PUTCHAR
12401                                                                     ; AL = character to display
12402 00000204 E2F8                               loop    con_lp
```

```
12403                                   cc_ret:
12404 00000206 F8                                 clc
12405 00000207 C3                                 retn
12406
12407                                   ; ============== S U B   R O U T I N E =======================================
12408
12409                                   ;****************************************************************************
12410                                   ;*                                                                          *
12411                                   ;*  con_flush - flush out keyboard queue                                    *
12412                                   ;*                                                                          *
12413                                   ;****************************************************************************
12414
12415                                   con_flush:
12416 00000208 C606[0C00]00                       mov     byte [altah], 0       ; clear out holding buffer
12417                                   flloop:                                 ; while (charavail()) charread();
12418 0000020D B401                               mov     ah, 1
12419 0000020F CD16                               int     16h             ; KEYBOARD - CHECK BUFFER, DO NOT CLEAR
12420                                                                     ; Return: ZF clear if character      in buffer
12421                                                                     ; AH = scan code, AL = character
12422                                                                     ; ZF set if no character in buffer
12423 00000211 74F3                               jz      short cc_ret
12424 00000213 30E4                               xor     ah, ah
12425 00000215 CD16                               int     16h             ; KEYBOARD - READ CHAR FROM BUFFER, WAIT IF EMPTY
12426                                                                     ; Return: AH = scan code, AL = character
12427 00000217 EBF4                               jmp     short flloop
12428
12429                                   ; ---------------------------------------------------------------------------
12430
12431                                   ; 15/10/2022
12432
12433                                   ;****************************************************************************
12434                                   ;*                                                                          *
12435                                   ;*  some equates for rom bios printer i/o                                       *
12436                                   ;*                                                                          *
12437                                   ;****************************************************************************
12438
12439                                   ; ibm rom status bits (i don't trust them, neither should you)
12440                                   ; warning!!! the ibm rom does not return just one bit. it returns a
12441                                   ; whole slew of bits, only one of which is correct.
12442
12443                                   ;notbusystatus     equ 10000000b         ; not busy
12444                                   ;nopaperstatus     equ 00100000b         ; no more paper
12445                                   ;prnselected       equ 00010000b         ; printer selected
12446                                   ;ioerrstatus       equ 00001000b         ; some kinda error
12447                                   ;timeoutstatus     equ 00000001b         ; time out.
12448                                   ;
12449                                   ;noprinter  equ 00110000b         ; no printer attached
12450
12451                                   ; 18/03/2019 - Retro DOS v4.0
12452                                   ;error_I24_out_of_paper   equ 9 ; MSDOS 6.0, ERR.INC, 1991
12453
12454                                   ; ---------------------------------------------------------------------------
12455
12456                                   ;****************************************************************************
12457                                   ;*                                                                          *
12458                                   ;*  prn_input - return with no error but zero chars read                        *
12459                                   ;*                                                                          *
12460                                   ;*  enter with cx = number of characters requested                          *
12461                                   ;*                                                                          *
12462                                   ;****************************************************************************
12463
12464                                   prn_input:                        ; 2C7h:21Ah = 70h:278Ah
12465 00000219 E8BBFE                             call    bc_err_cnt      ; reset count to zero
12466                                                                     ; (sub reqpkt.count,cx)
12467                                             ; 12/12/2022
12468                                   prn_done:
12469 0000021C F8                                 clc                     ; but return with carry      reset for no error
12470 0000021D C3                                 retn
12471                                   ; ---------------------------------------------------------------------------
12472
12473                                   ;****************************************************************************
12474                                   ;*                                                                          *
12475                                   ;*  prn_writ - write cx bytes from es:di to printer device                      *
12476                                   ;*                                                                          *
12477                                   ;*  auxnum has printer number                                               *
12478                                   ;*                                                                          *
12479                                   ;****************************************************************************
12480
12481                                   prn_writ:                         ; 2C7h:21Fh = 70h:278Fh
12482                                             ;jcxz  short prn_done ; no chars to output
12483 0000021E E3FC                               jcxz    prn_done        ; 19/10/2022
12484                                   prn_loop:
12485 00000220 BB0200                             mov     bx, 2           ; retry count
12486                                   prn_out:
12487 00000223 E83600                             call    prnstat         ; get status
12488 00000226 751D                               jnz     short TestPrnError
12489 00000228 268A05                             mov     al, [es:di]     ; get character       to print
12490 0000022B 30E4                               xor     ah, ah
12491 0000022D E82E00                             call    prnop           ; print to printer
12492 00000230 7419                               jz      short prn_con   ; no error - continue
12493 00000232 80FCFF                             cmp     ah, 0FFh        ; MODE_CTRLBRK
12494 00000235 7509                               jnz     short _prnwf
12495 00000237 B00C                               mov     al, 0Ch         ; error_I24_gen_failure
12496 00000239 C606[0C00]00                       mov     byte [altah], 0
12497 0000023E EB08                               jmp     short pmessg
12498                                   ; ---------------------------------------------------------------------------
12499
12500                                   _prnwf:
12501 00000240 F6C401                             test    ah, 1           ; timeoutstatus
12502 00000243 7406                               jz      short prn_con
12503                                   TestPrnError:
12504 00000245 4B                                 dec     bx              ; retry until count is exhausted.
12505 00000246 75DB                               jnz     short prn_out
12506                                   pmessg:
12507 00000248 E98CFE                             jmp     bc_err_cnt
12508                                   ; ---------------------------------------------------------------------------
12509
12510                                   prn_con:
12511 0000024B 47                                 inc     di              ; point to next char and continue
12512 0000024C E2D2                               loop    prn_loop
12513                                   ;prn_done:
12514                                             ; 12/12/2022
12515                                   prn_done2:
12516                                             ;clc
12517                                             ; cf=0
12518 0000024E C3                                 retn
12519                                   ; ---------------------------------------------------------------------------
12520
12521                                   ;****************************************************************************
12522                                   ;*                                                                          *
12523                                   ;*  prn_stat - device driver entry to return printer status                 *
12524                                   ;*                                                                          *
12525                                   ;****************************************************************************
12526
```

```
                        prn_stat:                       ; 2C7h:251h = 70h:27C1h
0000024F E80A00                 call    prnstat         ; device in dx
00000252 75F4                   jnz     short pmessg
00000254 F6C480                 test    ah, 80h         ; notbusystatus
                                ;jnz    short prn_done
                                ; 12/12/2022
00000257 75F5                   jnz     short prn_done2 ; cf=0
00000259 E96BFF                 jmp     z_bus_exit
                        ; ---------------------------------------------------------------------------

                        ;*************************************************************************
                        ;*                                                                       *
                        ;*  prnstat - utility function to call ROM BIOS to check                *
                        ;*           printer status. Return meaningful error code                *
                        ;*                                                                       *
                        ;*************************************************************************

                        prnstat:
0000025C B402                   mov     ah, 2           ; set command for get status
                                                        ; PRINTER - GET       STATUS
                                                        ; DX = printer port (0-3)
                                                        ; Return: AH = status

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

                        ;*************************************************************************
                        ;*                                                                       *
                        ;*  prnop - call ROM BIOS printer function in ah                         *
                        ;*          return zero true if no error                                 *
                        ;*          return zero false if error, al = error code                  *
                        ;*                                                                       *
                        ;*************************************************************************

                        prnop:
                                ; 20/12/2023 - Retro DOS v5.0
                                ; PCDOS 7.1 IBMBIO.COM

                                ;mov    dx, [auxnum]    ; get printer number
                                ;int    17h

0000025E 1E                     push    ds
0000025F FF36[2100]             push    word [auxnum]
00000263 31D2                   xor     dx, dx ; 0
00000265 8EDA                   mov     ds, dx
00000267 5A                     pop     dx
00000268 9C                     pushf                   ; simulate int 17h
00000269 FA                     cli
                                ;call   dword ptr ds:5ghghCh
0000026A FF1E5C00               call    far [17h*4]     ; 0:5Ch = INT 17h vector
0000026E 1F                     pop     ds

                            ; This check was added to see if this is a case of no
                            ; printer being installed. This tests checks to be sure
                            ; the error is noprinter (30h)

0000026F 50                         push    ax
00000270 80E430                     and     ah, 30h
00000273 80FC30                     cmp     ah, 30h         ; noprinter
00000276 58                         pop     ax
00000277 7506                       jnz     short NextTest
00000279 80E4DF                     and     ah, 0DFh        ; ~nopaperstatus
0000027C 80CC08                     or      ah, 8           ; ioerrstatus

                        ; examine the status bits to see if an error occurred. unfortunately, several
                        ; of the bits are set so we have to pick and choose. we must be extremely
                        ; careful about breaking basic.

                        NextTest:
0000027F F6C428                 test    ah, 28h         ; (ioerrstatus+nopaperstatus)
                                                        ; i/o error?
00000282 740A                   jz      short checknotready ; no, try not ready

                        ; at this point, we know we have an error. the converse is not true

00000284 B009                   mov     al, 9           ; error_I24_out_of_paper
                                                        ; first, assume       out of paper
00000286 F6C420                 test    ah, 20h         ; out of paper set?
00000289 7502                   jnz     short ret1      ; yes, error is       set
0000028B FEC0                   inc     al              ; return al=10 (i/o error)
                        ret1:
0000028D C3                     retn
                        ; ---------------------------------------------------------------------------

                        checknotready:
0000028E B002                   mov     al, 2           ; assume not-ready
00000290 F6C401                 test    ah, 1
00000293 C3                     retn

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

                        ;*************************************************************************
                        ;*                                                                       *
                        ;*  prn_tilbusy - output until busy. Used by print spooler.              *
                        ;*                this entry point should never block waiting for        *
                        ;*                device to come ready.                                  *
                        ;*                                                                       *
                        ;*  inputs: cx = count, es:di -> buffer                                  *
                        ;*  outputs: set the number of bytes transferred in the                  *
                        ;*           device driver request packet                                *
                        ;*                                                                       *
                        ;*************************************************************************

                                ; 19/10/2022
                        prn_tilbusy:                            ; 2C7h:28Bh = 70h:27FBh
00000294 89FE                   mov     si, di          ; everything is       set for lodsb
                        prn_tilbloop:
00000296 51                     push    cx
00000297 53                     push    bx
00000298 30FF                   xor     bh, bh
0000029A 8A1E[8004]             mov     bl, [printdev]
0000029E D1E3                   shl     bx, 1
                                ;mov    cx, ds:wait_count[bx] ;       wait count times to come ready
000002A0 8B8F[8104]             mov     cx, [wait_count+bx]
000002A4 5B                     pop     bx
                        prn_getstat:
000002A5 E8B4FF                 call    prnstat         ; get status
000002A8 7514                   jnz     short prn_bperr         ; error
000002AA F6C480                 test    ah, 80h         ; ready yet?
000002AD E1F6                   loope   prn_getstat     ; no, go for more
000002AF 59                     pop     cx              ; get original count
000002B0 740D                   jz      short prn_berr ; still not ready => done
000002B2 26                     es
000002B3 AC                     lodsb
                                ;lods   byte ptr es:[si] ; es
```

```
                                                        ; lodsb
000002B4 30E4                           xor     ah, ah
000002B6 E8A5FF                         call    prnop
000002B9 7504                           jnz     short prn_berr ; error
000002BB E2D9                           loop    prn_tilbloop
                                        ; 12/12/2022
                                        ; cf=0 (prnop)
                                        ;clc                    ; normal no-error return
000002BD C3                             retn                    ;   from device driver

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

                        prn_bperr:
000002BE 59                             pop     cx              ; restore transfer count from stack
                        prn_berr:
000002BF E915FE                         jmp     bc_err_cnt
                        ; ---------------------------------------------------------------------------

                        ; 15/10/2022

                        ;****************************************************************************
                        ;*                                                                          *
                        ;*  prn_genioctl - get/set printer retry count                              *
                        ;*                                                                          *
                        ;****************************************************************************

                        ; IOCTL.INC (MSDOS 6.0, 1991)
                        ; 11/01/2019

                        ;*******************************;*
                        ; CHARACTER DEVICES (PRINTERS)    ;*
                        ;*******************************;*

                        ;;RAWIO SUB-FUNCTIONS
                        ;;get_retry_count equ 65h
                        ;;set_retry_count equ 45h

                        ;;struc A_RETRYCOUNT
                        ;;.rc_count: resw 1
                        ;;endstruc

                        ;ioc_pc equ 5

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

                                        ; 19/10/2022
                        prn_genioctl:                           ; 2C7h:2BAh = 70h:282Ah
000002C2 C43E[1200]                     les     di, [ptrsav]
000002C6 26807D0D05                     cmp     byte [es:di+13], 5 ; [es:di+IOCTL_REQ.MAJORFUNCTION]
                                                                ; ioc_pc
000002CB 7403                           jz      short prnfunc_ok

                        prnfuncerr:
000002CD E905FE                         jmp     bc_cmderr
                        ; ---------------------------------------------------------------------------

                        prnfunc_ok:
000002D0 268A450E                       mov     al, [es:di+14] ; [es:di+IOCTL_REQ.MINORFUNCTION]
000002D4 26C47D13                       les     di, [es:di+19] ; [es:di+IOCTL_REQ.GENERICIOCTL_PACKET]
000002D8 30FF                           xor     bh, bh
                                        ;mov    bl, ds:printdev         ; get index into retry counts
000002DA 8A1E[8004]                     mov     bl, [printdev]
000002DE D1E3                           shl     bx, 1
                                        ;mov    cx, ds:wait_count[bx] ;       pull out retry count for device
000002E0 8B8F[8104]                     mov     cx, [wait_count+bx]
000002E4 3C65                           cmp     al, 65h         ; get_retry_count
000002E6 7407                           jz      short prngetcount
000002E8 3C45                           cmp     al, 45h         ; set_retry_count
000002EA 75E1                           jnz     short prnfuncerr
000002EC 268B0D                         mov     cx, [es:di]
                        prngetcount:
                                        ;mov    ds:wait_count[bx], cx
000002EF 898F[8104]                     mov     [wait_count+bx], cx
000002F3 26890D                         mov     [es:di], cx     ; [es:di+A_RETRYCOUNT.RC_COUNT]
                                                                ; return current retry count
                                        ; 12/12/2022
                                        ; cf=0
                                        ;clc
000002F6 C3                             retn
                        ; ---------------------------------------------------------------------------

                        ;****************************************************************************
                        ;*                                                                          *
                        ;*  prn_ioctl_query                                                             *
                        ;*                                                                          *
                        ;*  Added for 5.00                                                          *
                        ;****************************************************************************

                        prn_ioctl_query:                        ; 2C7h:2F0h = 70h:2860h
000002F7 C43E[1200]                     les     di, [ptrsav]
000002FB 26807D0D05                     cmp     byte [es:di+13], 5 ; [es:di+IOCTL_REQ.MAJORFUNCTION]
                                                                ; ioc_pc
00000300 750D                           jnz     short prn_query_err
00000302 268A450E                       mov     al, [es:di+14] ; [es:di+IOCTL_REQ.MINORFUNCTION]
00000306 3C65                           cmp     al, 65h         ; GET_RETRY_COUNT
00000308 7404                           jz      short IOCtlSupported
0000030A 3C45                           cmp     al, 45h         ; SET_RETRY_COUNT
0000030C 7501                           jnz     short prn_query_err
                        IOCtlSupported:
                                        ; 12/12/2022
                                        ; cf=0
                                        ;clc
0000030E C3                             retn
                        ; ---------------------------------------------------------------------------

                        prn_query_err:
                                        ; 12/12/2022
                                        ;stc
0000030F E9C3FD                         jmp     bc_cmderr ; (bc_cmderr sets cf to 1)
                        ; ---------------------------------------------------------------------------

                        ;****************************************************************************
                        ;*                                                                          *
                        ;*  aux port driver code -- "aux" == "com1"                                 *
                        ;*                                                                          *
                        ;*  the device driver entry/dispatch code sets up auxnum to                 *
                        ;*  give the com port number to use (0=com1, 1=com2, 2=com3...)             *
                        ;*                                                                          *
                        ;****************************************************************************

                        ;   values in ah, requesting function of int 14h in rom bios

                        ;auxfunc_send       equ   1      ;transmit
                        ;auxfunc_receive equ      2      ;read
```

```
12775                                   ;auxfunc_status     equ   3       ;request status
12776
12777                                   ;   error flags, reported by int 14h, reported in ah:
12778
12779                                   ;flag_data_ready equ      01h     ;data ready
12780                                   ;flag_overrun       equ   02h     ;overrun error
12781                                   ;flag_parity        equ   04h     ;parity error
12782                                   ;flag_frame  equ    08h     ;framing error
12783                                   ;flag_break  equ    10h     ;break detect
12784                                   ;flag_tranhol_emp equ     20h     ;transmit holding register empty
12785                                   ;flag_timeout       equ   80h     ;timeout
12786
12787                                   ;   these flags reported in al:
12788
12789                                   ;flag_cts    equ    10h     ;clear to send
12790                                   ;flag_dsr    equ    20h     ;data set ready
12791                                   ;flag_rec_sig       equ    80h     ;receive line signal detect
12792
12793                                   ; ---------------------------------------------------------------------------
12794
12795                                   ;*************************************************************************
12796                                   ;*                                                                       *
12797                                   ;*  aux_read - read cx bytes from [auxnum] aux port to buffer  *
12798                                   ;*             at es:di                                                  *
12799                                   ;*                                                                       *
12800                                   ;*************************************************************************
12801
12802                                   aux_read:                             ; 2C7h:30Dh = 70h:287Dh
12803                                             ;jcxz  short exvec2
12804 00000312 E311                               jcxz   exvec2         ; 19/10/2022
12805 00000314 E88000                             call   getbx          ; put address of auxbuf      in bx
12806 00000317 30C0                               xor    al, al
12807 00000319 8607                               xchg   al, [bx]
12808 0000031B 08C0                               or     al, al
12809 0000031D 7503                               jnz    short aux2
12810                                   aux1:
12811 0000031F E80500                             call   auxin          ; get character       from port
12812                                                                   ; won't return if error
12813                                   aux2:
12814 00000322 AA                                 stosb
12815 00000323 E2FA                               loop   aux1           ; if more characters, go around      again
12816                                   exvec2:
12817 00000325 F8                                 clc                   ; all done, successful exit
12818                                   auxin_retn: ; 18/12/2022
12819 00000326 C3                                 retn
12820                                   ; ---------------------------------------------------------------------------
12821
12822                                   ;*************************************************************************
12823                                   ;*                                                                       *
12824                                   ;*  auxin - call rom bios to read character from aux port                *
12825                                   ;*          if error occurs, map the error and return one                *
12826                                   ;*          level up to device driver exit code, setting         *
12827                                   ;*          the number of bytes transferred appropriately                *
12828                                   ;*                                                                       *
12829                                   ;*************************************************************************
12830
12831                                   auxin:
12832 00000327 B402                               mov    ah, 2          ; auxfunc_receive
12833 00000329 E83A00                             call   auxop
12834 0000032C F6C40E                             test   ah, 0Eh        ; flag_frame|flag_parity|flag_overrun
12835                                             ;jnz   short arbad    ; skip if any error bits set
12836                                             ;retn
12837                                             ; 25/06/2023 (BugFix)
12838 0000032F 74F5                               jz     short auxin_retn
12839                                   ; ---------------------------------------------------------------------------
12840
12841                                   arbad:
12842 00000331 58                                 pop    ax             ; remove return       address (near call)
12843                                             ;xor   al, al
12844                                             ;or    al, 0B0h       ; flag_rec_sig|       flag_dsr|flag_cts
12845                                             ; 11/08/2023
12846 00000332 B0B0                               mov    al, 0B0h       ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0334h
12847 00000334 E9A0FD                             jmp    bc_err_cnt
12848
12849                                   ; ---------------------------------------------------------------------------
12850
12851                                   ;*************************************************************************
12852                                   ;*                                                                       *
12853                                   ;*  aux_rdnd - non-destructive aux port read                             *
12854                                   ;*                                                                       *
12855                                   ;*************************************************************************
12856
12857                                   aux_rdnd:                             ; 2C7h:335h = 70h:28A5h
12858 00000337 E85D00                             call   getbx
12859 0000033A 8A07                               mov    al, [bx]       ; have bx point       to auxbuf
12860 0000033C 08C0                               or     al, al         ; if al is non-zero (char in buffer)
12861 0000033E 7511                               jnz    short auxdrx   ; then return character
12862 00000340 E82100                             call   auxstat        ; if not, get status of      aux device
12863 00000343 F6C401                             test   ah, 1          ; flag_data_ready - test data ready
12864 00000346 740C                               jz     short auxbus   ; then device is busy (not ready)
12865 00000348 A820                               test   al, 20h        ; flag_dsr - test data set ready
12866 0000034A 7408                               jz     short auxbus   ; then device is busy (not ready)
12867 0000034C E8D8FF                             call   auxin          ; else aux is ready, get character
12868 0000034F 8807                               mov    [bx], al
12869                                   auxdrx:
12870 00000351 E99EFE                             jmp    rdexit         ; return busy status
12871                                   ; ---------------------------------------------------------------------------
12872
12873                                   auxbus:
12874 00000354 E970FE                             jmp    z_bus_exit
12875                                   ; ---------------------------------------------------------------------------
12876
12877                                   ;*************************************************************************
12878                                   ;*                                                                       *
12879                                   ;*  aux_wrst - return aux port write status                                  *
12880                                   ;*                                                                       *
12881                                   ;*************************************************************************
12882
12883                                   aux_wrst:                             ; 2C7h:355h = 70h:28C5h
12884 00000357 E80A00                             call   auxstat        ; get status of       aux in ax
12885 0000035A A820                               test   al, 20h        ; test data set       ready
12886 0000035C 74F6                               jz     short auxbus   ; then device is busy (not ready)
12887 0000035E F6C420                             test   ah, 20h        ; flag_tranhol_emp - test transmit hold     reg empty
12888 00000361 74F1                               jz     short auxbus   ; then device is busy (not ready)
12889                                             ; 12/12/2022
12890                                             ; cf=0  ; (test instruction resets cf)
12891                                             ;clc
12892 00000363 C3                                 retn
12893                                   ; ---------------------------------------------------------------------------
12894
12895                                   ;*************************************************************************
12896                                   ;*                                                                       *
12897                                   ;*  auxstat - call rom bios to determine aux port status                 *
12898                                   ;*                                                                       *
```

```
12899                                   ;*  exit:   ax = status                                         *
12900                                   ;*          dx = [auxnum]                                       *
12901                                   ;*                                                              *
12902                                   ;****************************************************************************
12903                                   
12904                                   auxstat:
12905 00000364 B403                               mov     ah, 3           ; auxfunc_status
12906                                   
12907                                             ; fall into auxop
12908                                   
12909                                   ; =============== S U B   R O U T I N E =======================================
12910                                   
12911                                   ;****************************************************************************
12912                                   ;*                                                              *
12913                                   ;*  auxop - perform rom-biox aux port interrupt                 *
12914                                   ;*                                                              *
12915                                   ;*  entry:  ah = int 14h function number                        *
12916                                   ;*  exit:   ax = results                                        *
12917                                   ;*          dx = [auxnum]                                       *
12918                                   ;*                                                              *
12919                                   ;****************************************************************************
12920                                   
12921                                   auxop:            ; proc near
12922                                             ; 20/12/2023 - Retro DOS v5.0
12923                                             ;mov    dx, [auxnum]    ; ah=function code
12924                                             ;                       ; 0=init, 1=send, 2=receive, 3=status
12925                                             ;                       ; get port number
12926                                             ;
12927                                             ;int    14h             ; SERIAL I/O - GET USART STATUS
12928                                             ;                       ; DX = port number (0-3)
12929                                             ;                       ; Return: AX = port status code
12930                                             ; (PCDOS 7.1 IBMBIO.COM)
12931 00000366 1E                               push    ds
12932 00000367 FF36[2100]                       push    word [auxnum]
12933 0000036B 31D2                             xor     dx, dx ; 0
12934 0000036D 8EDA                             mov     ds, dx
12935 0000036F 5A                               pop     dx
12936 00000370 9C                               pushf                   ; simulate INT 14h
12937 00000371 FA                               cli
12938                                           ;call   dword ptr ds:50h
12939 00000372 FF1E5000                         call    far [14h*4]     ; INT 14h vector (14h*4 = 50h)
12940 00000376 1F                               pop     ds
12941 00000377 C3                               retn
12942                                   
12943                                   ; ---------------------------------------------------------------------------
12944                                   
12945                                   ;****************************************************************************
12946                                   ;*                                                              *
12947                                   ;*  aux_flsh - flush aux input buffer - set contents of         *
12948                                   ;*             auxbuf [auxnum] to zero                          *
12949                                   ;*                                                              *
12950                                   ;*  cas - shouldn't this code call the rom bios input function *
12951                                   ;*        repeatedly until it isn't ready?  to flush out any    *
12952                                   ;*        pending serial input queue if there's a tsr like MODE       *
12953                                   ;*        which is providing interrupt-buffering of aux port?   *
12954                                   ;*                                                              *
12955                                   ;****************************************************************************
12956                                   
12957                                   aux_flsh:                         ; 2C7h:36Ch = 70h:28DCh
12958 00000378 E81C00                             call    getbx           ; flush aux input buffer
12959 0000037B C60700                             mov     byte [bx], 0    ; get bx to point to auxbuf
12960                                                                     ; zero out buffer
12961                                             ;clc                    ; all done, successful return
12962                                             ; 12/12/2022
12963                                             ; cf=0 ('add' instruction in 'getbx')
12964 0000037E C3                               retn
12965                                   ; ---------------------------------------------------------------------------
12966                                   
12967                                   ;****************************************************************************
12968                                   ;*                                                              *
12969                                   ;*  aux_writ - write to aux device                                      *
12970                                   ;*                                                              *
12971                                   ;****************************************************************************
12972                                   
12973                                   aux_writ:                         ; 2C7h:374h = 70h:28E4h
12974                                             ;jcxz   short exvec2    ; write to aux device (if cx > 0)
12975 0000037F E3A4                             jcxz    exvec2          ; 19/10/2022
12976                                   aux_loop:
12977 00000381 268A05                           mov     al, [es:di]     ; get character       to be written
12978                                                                     ; move di pointer to next character
12979 00000384 47                               inc     di
12980 00000385 B401                             mov     ah, 1           ; auxfunc_send - indicates a write
12981 00000387 E8DCFF                           call    auxop           ; send character over aux port
12982 0000038A F6C480                           test    ah, 80h         ; check for error
12983 0000038D 7405                             jz      short awok      ; then no error
12984 0000038F B00A                             mov     al, 10          ; else indicate       write fault
12985 00000391 E943FD                           jmp     bc_err_cnt      ; call error routines
12986                                   ; ---------------------------------------------------------------------------
12987                                   
12988                                   awok:
12989 00000394 E2EB                             loop    aux_loop        ; if cx is non-zero,
12990                                                                     ; still more character to print
12991                                             ;clc                    ; all done, successful return
12992                                             ; 12/12/2022
12993                                             ; cf=0 (test instruction above)
12994 00000396 C3                               retn
12995                                   
12996                                   ; =============== S U B   R O U T I N E =======================================
12997                                   
12998                                   ;****************************************************************************
12999                                   ;*                                                              *
13000                                   ;*  getbx - return bx -> single byte input buffer for           *
13001                                   ;*          selected aux port ([auxnum])                        *
13002                                   ;*                                                              *
13003                                   ;****************************************************************************
13004                                   
13005                                   getbx:
13006 00000397 8B1E[2100]                       mov     bx, [auxnum]    ; return bx -> single byte input buffer
13007                                                                     ; for selected aux port       ([auxnum])
13008                                             ;add    bx, offset auxbuf
13009 0000039B 81C3[1600]                       add     bx, auxbuf      ; 19/10/2022
13010                                           ; 12/12/2022
13011                                           ; cf=0 (if [auxnum] is valid number)
13012 0000039F C3                               retn
13013                                   
13014                                   ; ---------------------------------------------------------------------------
13015                                   
13016                                   ; 15/10/2022
13017                                   
13018                                   ;--------------------------------------------------------------
13019                                   ;                                                             :
13020                                   ;              clock device driver                            :
13021                                   ;                                                             :
13022                                   ;                                                             :
```

```
13023                                   ;   this file contains the clock device driver.        :
13024                                   ;                                                         :
13025                                   ;   the routines in this files are:                          :
13026                                   ;                                                         :
13027                                   ;   routine               function                    :
13028                                   ;   -------               --------                    :
13029                                   ;   tim_writ              set the current time        :
13030                                   ;   tim_read              read the current time       :
13031                                   ;   time_to_ticks         convert time to corresponding:
13032                                   ;                             number of clock ticks   :
13033                                   ;                                                         :
13034                                   ; the clock ticks at the rate of:                         :
13035                                   ;                                                         :
13036                                   ;   1193180/65536 ticks/second (about 18.2 ticks per second):
13037                                   ; see each routine for information on the use.                :
13038                                   ;                                                         :
13039                                   ;---------------------------------------------------------------
13040
13041                                   ; convert time to ticks
13042                                   ; input : time in cx and dx
13043                                   ; ticks returned in cx:dx
13044
13045                                   ;19/07/2019
13046                                   ;09/03/2019
13047
13048                                   time_to_ticks:                          ; 0070h:2906h =       02C7h:0396h
13049
13050                                   ; first convert from hour,min,sec,hund. to
13051                                   ; total number of 100th of seconds
13052
13053 000003A0 B03C                                mov     al, 60
13054 000003A2 F6E5                                mul     ch              ; hours to minutes
13055 000003A4 B500                                mov     ch, 0
13056 000003A6 01C8                                add     ax, cx          ; total minutes
13057 000003A8 B97017                              mov     cx, 6000        ; 60*100
13058 000003AB 89D3                                mov     bx, dx          ; get out of the way of      the multiply
13059 000003AD F7E1                                mul     cx              ; convert to 1/100 sec
13060 000003AF 89C1                                mov     cx, ax
13061 000003B1 B064                                mov     al, 100
13062 000003B3 F6E7                                mul     bh              ; convert seconds to 1/100 sec
13063 000003B5 01C1                                add     cx, ax          ; combine seconds with hours and min
13064 000003B7 83D200                              adc     dx, 0           ; ripple carry
13065 000003BA B700                                mov     bh, 0
13066 000003BC 01D9                                add     cx, bx          ; combine 1/100       sec
13067 000003BE 83D200                              adc     dx, 0
13068
13069                                                 ; dx:cx is time in 1/100 sec
13070
13071 000003C1 92                                  xchg    ax, dx
13072 000003C2 91                                  xchg    ax, cx          ; now time is in cx:ax
13073 000003C3 BB0BE9                              mov     bx, 59659
13074 000003C6 F7E3                                mul     bx              ; multiply low half
13075 000003C8 87CA                                xchg    dx, cx
13076 000003CA 92                                  xchg    ax, dx          ; cx->ax, ax->dx, dx->cx
13077 000003CB F7E3                                mul     bx              ; multiply high       half
13078 000003CD 01C8                                add     ax, cx          ; combine overlapping products
13079 000003CF 83D200                              adc     dx, 0
13080 000003D2 92                                  xchg    ax, dx          ; ax:dx=time*59659
13081 000003D3 BB0500                              mov     bx, 5
13082 000003D6 F6F3                                div     bl              ; divide high half by 5
13083 000003D8 88C1                                mov     cl, al
13084 000003DA B500                                mov     ch, 0
13085 000003DC 88E0                                mov     al, ah          ; remainder of divide-by-5
13086 000003DE 98                                  cbw
13087 000003DF 92                                  xchg    ax, dx          ; use it to extend low half
13088 000003E0 F7F3                                div     bx              ; divide low half by 5
13089 000003E2 89C2                                mov     dx, ax          ; cx:dx is now number of ticks in time
13090 000003E4 CB                                  retf                    ; far return
13091
13092                                   ; ----------------------------------------------------------------------------
13093
13094                                   ; 17/10/2022
13095                                   ; 15/10/2022
13096
13097                                   ;---------------------------------------------------------------
13098                                   ;
13099                                   ; tim_writ sets the current time
13100                                   ;
13101                                   ; on entry es:[di] has the current time:
13102                                   ;
13103                                   ;   number of days since 1-1-80   (word)
13104                                   ;   minutes (0-59)                (byte)
13105                                   ;   hours (0-23)                  (byte)
13106                                   ;   hundredths of seconds (0-99)  (byte)
13107                                   ;   seconds (0-59)                (byte)
13108                                   ;
13109                                   ; each number has been checked for the correct range.
13110                                   ;
13111                                   ;   NOTE: Any changes in this routine probably require corresponding
13112                                   ;   changes in the version that is built with the power manager driver.
13113                                   ;   See ptime.asm.
13114                                   ;
13115                                   ;---------------------------------------------------------------
13116
13117                                       ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
13118                                       ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:03EAh
13119                                   tim_writ:                               ; 2C7h:3DBh = 70h:294Bh
13120 000003E5 268B05                             mov     ax, [es:di]
13121 000003E8 50                                  push    ax              ; daycnt. we need to set this at the very
13122                                                                      ; end to avoid tick windows.
13123 000003E9 803E[8C04]00                        cmp     byte [havecmosclock], 0
13124                                              ;cmp    ds:havecmosclock, 0
13125 000003EE 7423                                jz      short no_cmos_1
13126 000003F0 268A4503                            mov     al, [es:di+3]  ; near indirect       calls
13127                                                                      ; get binary hours
13128                                                                      ; convert to bcd
13129                                              ;call   far [bintobcd]
13130                                              ;;call  ds:bintobcd    ; call far [bintobcd]
13131                                              ; 08/08/2023
13132 000003F4 E8E800                              call    bintobcd
13133 000003F7 88C5                                mov     ch, al          ; ch = bcd hours
13134 000003F9 268A4502                            mov     al, [es:di+2]  ; get binary minutes
13135                                              ;call   far [bintobcd]
13136                                              ;;call  ds:bintobcd    ; convert to bcd
13137 000003FD E8DF00                              call    bintobcd
13138 00000400 88C1                                mov     cl, al          ; cl = bcd minutes
13139 00000402 268A4505                            mov     al, [es:di+5]  ; get binary seconds
13140                                              ;call   far [bintobcd]
13141                                              ;;call  ds:bintobcd
13142 00000406 E8D600                              call    bintobcd
13143
13144 00000409 88C6                                mov     dh, al          ; dh = bcd seconds
13145 0000040B B200                                mov     dl, 0           ; dl = 0 (st) or 1 (dst)
13146 0000040D FA                                  cli
```

```
13147 0000040E B403                             mov     ah, 3
13148 00000410 CD1A                             int     1Ah             ; CLOCK - SET REAL TIME       CLOCK (AT,XT286,CONV,PS)
13149                                                                   ; CH = hours in       BCD, CL = minutes in BCD
13150                                                                   ; DH = seconds in BCD
13151                                                                   ; DL = 01h if daylight savings, 00h if standard time
13152                                                                   ; Return: CMOS clock set
13153 00000412 FB                               sti
13154                                   no_cmos_1:
13155 00000413 268B4D02                         mov     cx, [es:di+2]
13156 00000417 268B5504                         mov     dx, [es:di+4]
13157                                           ; 17/10/2022
13158 0000041B FF1E[0606]                       call    far [ttticks]
13159                                           ;call   dword ptr ds:ttticks ; call far      [ttticks]
13160                                                                   ; convert time to ticks
13161                                                                   ; cx:dx now has time in ticks
13162 0000041F FA                               cli                     ; turn off timer
13163 00000420 B401                             mov     ah, 1
13164 00000422 CD1A                             int     1Ah             ; CLOCK - SET TIME OF DAY
13165                                                                   ; CX:DX = clock count
13166                                                                   ; Return: time of day set
13167                                           ;pop    ds:daycnt
13168 00000424 8F06[8904]                       pop     word [daycnt]
13169 00000428 FB                               sti
13170                                           ;cmp    ds:havecmosclock, 0
13171 00000429 803E[8C04]00                     cmp     byte [havecmosclock], 0
13172 0000042E 7409                             jz      short no_cmos_2
13173
13174                                           ; 08/08/2023
13175                                           ;call   far [daycnttoday]
13176                                           ;;call  ds:daycnttoday ; call far [daycnttoday]
13177                                                                   ; convert to bcd format
13178 00000430 E80700                           call    daycnttoday
13179
13180 00000433 FA                               cli
13181 00000434 B405                             mov     ah, 5
13182 00000436 CD1A                             int     1Ah             ; CLOCK - SET DATE IN REAL TIME       CLOCK (AT,XT286,CONV,PS)
13183                                                                   ; DL = day in BCD, DH =       month in BCD, CL = year       in BCD
13184                                                                   ; CH = century (19h or 20h)
13185                                                                   ; Return: CMOS clock set
13186 00000438 FB                               sti
13187                                   no_cmos_2:
13188                                                   ; 12/12/2022
13189                                                   ; cf=0
13190                                                   ;clc
13191 00000439 C3                                       retn
13192
13193                                   ; ---------------------------------------------------------------------------
13194
13195                                   ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
13196                                   ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0440h
13197                                   %if 1
13198
13199                                   ; CMOS Clock setting support routines used by MSCLOCK.
13200                                   ; Warning!!! This code will be dynamically relocated by MSINIT.
13201
13202                                   daycnttoday:       ; proc near
13203
13204                                   ; entry: [daycnt] = number of days since 1-1-80
13205                                   ;
13206                                   ; return: ch - century in bcd
13207                                   ;     cl - year in bcd
13208                                   ;     dh - month in bcd
13209                                   ;     dl - day in bcd
13210
13211                                                   ; 20/12/2023 - Retro DOS v5.0
13212
13213                                                   ; 08/08/2023 (ds:) (near proc)
13214                                                   ; 16/10/2022 (cs:) (far proc)
13215 0000043A FF36[8904]                               push    word [daycnt] ; save daycnt
13216 0000043E 813E[8904]891C                           cmp     word [daycnt], 7305 ; (365*20+(20/4))
13217                                                                   ; # days from 1-1-1980 to 1-1-2000
13218 00000444 7308                                     jnb     short century20
13219                                                   ;mov    byte [base_century], 19
13220                                                   ;mov    byte [base_year], 80
13221                                                   ; 08/08/2023
13222 00000446 C706[8D04]1350                           mov     word [base_century], 5013h
13223 0000044C EB0C                                     jmp     short years
13224                                   ; ---------------------------------------------------------------------------
13225
13226                                   century20:
13227                                                   ;mov    byte [base_century], 20
13228                                                   ;mov    byte [base_year], 0
13229                                                   ; 08/08/2023
13230 0000044E C706[8D04]1400                           mov     word [base_century], 20
13231 00000454 812E[8904]891C                           sub     word [daycnt], 7305 ; (365*20+(20/4))
13232                                                                   ; adjust daycnt
13233                                   years:
13234 0000045A 31D2                                     xor     dx, dx
13235 0000045C A1[8904]                                 mov     ax, [daycnt]
13236 0000045F BBB505                                   mov     bx, 1461        ; (366+365*3)
13237                                                                   ; # of days in a Leap year block
13238 00000462 F7F3                                     div     bx              ; AX = # of leap block,       DX = daycnt
13239 00000464 8916[8904]                               mov     [daycnt], dx    ; save daycnt left
13240 00000468 B304                                     mov     bl, 4
13241 0000046A F6E3                                     mul     bl              ; AX = # of years. Less       than 100
13242 0000046C 0006[8E04]                               add     [base_year], al ; So, ah = 0. Adjust year
13243 00000470 FF06[8904]                               inc     word [daycnt]  ; set daycnt to       1 base
13244                                                   ; 08/08/2023
13245 00000474 BB6E01                                   mov     bx, 366
13246 00000477 B90300                                   mov     cx, 3
13247                                                   ;cmp    word [daycnt], 366 ; daycnt=remainder of leap year
13248 0000047A 391E[8904]                               cmp     [daycnt], bx ; 366
13249 0000047E 7619                                     jbe     short leapyear ; within 366+355+355+355 days.
13250 00000480 FE06[8E04]                               inc     byte [base_year] ; if daycnt <= 366, then leap year
13251                                                   ;sub    word [daycnt], 366 ; else daycnt--, base_year++ ;
13252 00000484 291E[8904]                               sub     [daycnt], bx ; 366 ; 08/08/2023
13253                                                   ;mov    cx, 3          ; And next three years are normal
13254                                                   ; 08/08/2023
13255 00000488 4B                                       dec     bx ; 365
13256                                   regularyear:            ; 20/12/2023
13257                                                   ;cmp    word [daycnt], 365 ; for(i=1; i>3 or daycnt <=365; i++)
13258 00000489 391E[8904]                               cmp     [daycnt], bx ; 365 ; 08/08/2023
13259 0000048D 760F                                     jbe     short yeardone ; {if (daycnt >       365)
13260 0000048F FE06[8E04]                               inc     byte [base_year] ; { daycnt -=       365
13261                                                   ;sub    word [daycnt], 365 ; }
13262 00000493 291E[8904]                               sub     [daycnt], bx ; 365 ; 08/08/2023
13263 00000497 E2F0                                     loop    regularyear    ; }
13264                                                                   ;
13265                                                                   ; should never fall through loop
13266                                   leapyear:
13267 00000499 C606[9004]1D                             mov     byte [february], 29 ; 08/08/2023
13268                                                   ;mov    byte [month_tab+1], 29 ; leap year.
13269                                                                   ; change month table.
13270                                   yeardone:
```

```
13271 0000049E 31DB                             xor     bx, bx
13272 000004A0 31D2                             xor     dx, dx
13273 000004A2 A1[8904]                         mov     ax, [daycnt]
13274                                           ;mov    si, offset month_tab
13275 000004A5 BE[8F04]                         mov     si, month_tab  ; 19/10/2022
13276                                           ;mov    cx, 12
13277                                           ; 08/08/2023
13278 000004A8 B10C                             mov     cl, 12
13279                                   months:
13280 000004AA FEC3                             inc     bl
13281                                           ; 08/08/2023
13282 000004AC 8A14                             mov     dl, [si]        ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:04B7h
13283 000004AE 39D0                             cmp     ax, dx          ; cmp daycnt for each month till fit
13284                                           ; dh=0
13285 000004B0 7605                             jbe     short month_done
13286 000004B2 46                               inc     si              ; next month
13287 000004B3 29D0                             sub     ax, dx          ; adjust daycnt
13288 000004B5 E2F3                             loop    months          ;
13289                                           ; should never fall through loop
13290                                   month_done:
13291 000004B7 C606[9004]1C                     mov     byte [february], 28 ; 08/08/2023
13292                                           ;mov    byte [month_tab+1], 28
13293                                           ; restore month table value
13294 000004BC 88DA                             mov     dl, bl
13295 000004BE 8A36[8E04]                       mov     dh, [base_year]
13296 000004C2 8A0E[8D04]                       mov     cl, [base_century] ; al=day,dl=month,dh=year,cl=cntry
13297 000004C6 E81600                           call    bintobcd        ; convert "day"       to bcd
13298                                           ; dl = bcd day,       al = month
13299 000004C9 86C2                             xchg    dl, al
13300 000004CB E81100                           call    bintobcd        ; dh = bcd month, al = year
13301 000004CE 86C6                             xchg    dh, al
13302 000004D0 E80C00                           call    bintobcd        ; cl = bcd year, al = century
13303 000004D3 86C1                             xchg    cl, al
13304 000004D5 E80700                           call    bintobcd        ; ch = bcd century
13305 000004D8 88C5                             mov     ch, al
13306 000004DA 8F06[8904]                       pop     word [daycnt] ; restore original value
13307 000004DE C3                               retn
13308
13309                                   ;---------------------------------------------------------------------------
13310
13311                                   bintobcd:   ; proc near           ; real time clock support
13312
13313                                   ;convert a binary input in al (less than 63h or 99 decimal)
13314                                   ;into a bcd value in al. ah destroyed.
13315
13316 000004DF D40A                               aam                     ; AH = AL/10, AL = AL MOD 10
13317 000004E1 D510                               aad     10h             ; db 0D5h,10h
13318                                             ; AL = (AH*10H)+AL, AH = 0
13319 000004E3 C3                               retn
13320                                   %endif
13321
13322                                   ;---------------------------------------------------------------------------
13323
13324                                   ; 15/10/2022
13325
13326                                   ;---------------------------------------------------------------------------
13327                                   ; gettime reads date and time
13328                                   ; and returns the following information:
13329                                   ;
13330                                   ;   es:[di]  =count of days since 1-1-80
13331                                   ;   es:[di+2]=hours
13332                                   ;   es:[di+3]=minutes
13333                                   ;   es:[di+4]=seconds
13334                                   ;   es:[di+5]=hundredths of seconds
13335                                   ;
13336                                   ;   NOTE: Any changes in this routine probably require corresponding
13337                                   ;   changes in the version that is built with the power manager driver.
13338                                   ;   See ptime.asm.
13339                                   ;---------------------------------------------------------------------------
13340
13341                                   tim_read:                         ; 2C7h:435h = 70h:29A5h
13342 000004E4 E84A00                             call    GetTickCnt
13343 000004E7 8B36[8904]                         mov     si, [daycnt]
13344
13345                                   ; we now need to convert the time in tick to the time in 100th of
13346                                   ; seconds. the relation between tick and seconds is:
13347                                   ;
13348                                   ;           65,536 seconds
13349                                   ;          ----------------
13350                                   ;          1,193,180 tick
13351                                   ;
13352                                   ; to get to 100th of second we need to multiply by 100. the equation is:
13353                                   ;
13354                                   ;   ticks from clock  * 65,536 * 100
13355                                   ;      --------------------------------- = time in 100th of seconds
13356                                   ;           1,193,180
13357                                   ;
13358                                   ; fortunately this formula simplifies to:
13359                                   ;
13360                                   ;   ticks from clock * 5 * 65,536
13361                                   ;      ---------------------------------- = time in 100th of seconds
13362                                   ;           59,659
13363                                   ;
13364                                   ; the calculation is done by first multipling tick by 5. next we divide by
13365                                   ; 59,659. in this division we multiply by 65,536 by shifting the dividend
13366                                   ; my 16 bits to the left.
13367                                   ;
13368                                   ; start with ticks in cx:dx
13369                                   ; multiply by 5
13370
13371 000004EB 89C8                               mov     ax, cx
13372 000004ED 89D3                               mov     bx, dx          ; start with ticks in cx:dx
13373                                             ; multiply by 5
13374 000004EF D1E2                               shl     dx, 1
13375 000004F1 D1D1                               rcl     cx, 1           ; times 2
13376 000004F3 D1E2                               shl     dx, 1
13377 000004F5 D1D1                               rcl     cx, 1           ; times 4
13378 000004F7 01DA                               add     dx, bx
13379 000004F9 11C8                               adc     ax, cx          ; times 5
13380 000004FB 92                                 xchg    ax, dx
13381
13382                                   ; now have ticks * 5 in   dx:ax
13383                                   ; we now need to multiply by 65536 and divide by 59659 d.
13384
13385 000004FC B9OBE9                             mov     cx, 59659       ; get divisor
13386 000004FF F7F1                               div     cx              ; dx now has remainder
13387                                             ; ax has high word of final quotient
13388
13389                                             ; 08/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
13390                                             ;mov    bx, ax          ; put high word       in safe place
13391 00000501 93                                 xchg    bx, ax
13392 00000502 31C0                               xor     ax, ax          ; this is the multiply by 65536
13393 00000504 F7F1                               div     cx              ; bx:ax now has time in 100th of seconds
13394
```

```
                                ; 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

                                %endif

                                ; 21/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
                                ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0579h

                                        ; 21/12/2023 - Retro DOS v5.0
                                        ; 11/12/2022
0000056F 1A                     dsktbl:         db (dtbl_siz-1)/2     ; 26 ; this is the size of the table
00000570 [4A1A]                 dw dsk_init
00000572 [B805]                 dw media_chk
00000574 [5706]                 dw get_bpb
                                ;dw bc_cmderr
00000576 [4A0E]                 dw ioctl_input ; PCDOS 7 ; 21/12/2023
00000578 [7209]                 dw dsk_read
0000057A [5809]                 dw x_bus_exit
0000057C [2206]                 dw ret_carry_clear
0000057E [2206]                 dw ret_carry_clear
00000580 [6409]                 dw dsk_writ
00000582 [5C09]                 dw dsk_writv
00000584 [2206]                 dw ret_carry_clear
00000586 [2206]                 dw ret_carry_clear
                                ;dw bc_cmderr
00000588 [F70D]                 dw ioctl_output ; PCDOS 7 ; 21/12/2023
0000058A [2909]                 dw dsk_open
0000058C [3809]                 dw dsk_close
0000058E [4E09]                 dw dsk_rem
00000590 [2206]                 dw ret_carry_clear
00000592 [2206]                 dw ret_carry_clear
00000594 [2206]                 dw ret_carry_clear
00000596 [CC0E]                 dw do_generic_ioctl
00000598 [2206]                 dw ret_carry_clear
0000059A [2206]                 dw ret_carry_clear
0000059C [2206]                 dw ret_carry_clear
0000059E [A713]                 dw ioctl_getown
000005A0 [C413]                 dw ioctl_setown
000005A2 [1C15]                 dw ioctl_support_query

                                dtbl_siz equ $-dsktbl

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

                                ; ---------------------------------------------------------------------------
                                ; setdrive scans through the data structure of bdss, and returns a pointer to
                                ; the one that belongs to the drive specified. carry is set if none exists
                                ; for the drive. Pointer is returned in es:[di]
                                ;
                                ;  AL contains the logical drive number.
                                ; ---------------------------------------------------------------------------

                                SetDrive:
                                        ;les    di, dword ptr ds:start_bds ; Point es:di to first bds
000005A4 C43E[1901]                     les     di, [start_bds] ; 19/10/2022
                                X_Scan_Loop:
000005A8 26384505                       cmp     [es:di+5], al
000005AC 7409                           jz      short X_SetDrv
000005AE 26C43D                         les     di, [es:di]    ; [es:di+BDS.link] ; Go      to next bds
000005B1 83FFFF                         cmp     di, 0FFFFh
000005B4 75F2                           jnz     short X_Scan_Loop
000005B6 F9                             stc
                                X_SetDrv:
000005B7 C3                             retn

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

                                ; 15/10/2022

                                    ; 21/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
                                    ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:05C2h

                                ; ---------------------------------------------------------------------
                                ; if id is f9, have a 96tpi disk else
                                ; if bit 2 is 0 then media is not removable and could not have changed
                                ;  otherwise if within 2 secs of last disk operation media could not
                                ;    have changed, otherwise dont know if media has changed
                                ; ---------------------------------------------------------------------

                                media_chk:                      ; 2C7h:4EBh = 70h:2A5Bh
000005B8 E8E9FF                         call    SetDrive
000005BB BE0100                         mov     si, 1
                                        ; 21/12/2023
000005BE 26F6454001                     test    byte [es:di+40h], 1
                                        ;test   byte [es:di+24h], 1 ; [es:di+BDS.flags+1]
                                                                    ; fchanged_by_format
000005C3 7415                           jz      short WeAreNotFakingIt
                                        ; 21/12/2023
000005C5 26806540FE                     and     byte [es:di+40h], 0FEh
                                        ; 12/12/2022
                                        ;and    byte [es:di+24h], 0FEh ; ~fchanged_by_format
                                        ;;and   word [es:di+23h], 0FEFFh ; [es:di+BDS.flags]
                                                                    ; ~fchanged_by_format ;       reset flag
000005CA C606[1E01]FF                   mov     byte [tim_drv], 0FFh ; -1
                                                                    ; Ensure that we ask the rom if media has changed
                                        ; 21/12/2023
000005CF 26F6453F01                     test    byte [es:di+3Fh], 1
                                        ;test   byte [es:di+23h], 1 ; [es:di+BDS.flags]
                                                                    ; fnon_removable
000005D4 740B                           jz      short wehaveafloppy
                                        ;mov    si, 0FFFFh      ; Indicate media changed
                                        ; 11/08/2023
000005D6 F7DE                           neg     si              ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:05E0h
000005D8 EB2B                           jmp     short Media_Done ; Media_Done
                                ; ---------------------------------------------------------------------------

                                WeAreNotFakingIt:
                                        ; 21/12/2023
000005DA 26F6453F01                     test    byte [es:di+3Fh], 1
                                        ;test   byte [es:di+BDS.flags], fnon_removable
                                        ;test   byte [es:di+23h], 1
000005DF 7524                           jnz     short Media_Done
                                wehaveafloppy:
                                        ;xor    si, si ; 0      ; Presume "I don't know"
                                        ; 11/08/2023
000005E1 4E                             dec     si ; 0          ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:05EBh

                                        ; If we have a floppy with changeline support, we ask the ROM
                                        ; to determine if media has changed. We do not perform the
                                        ; 2 second check for these drives.

000005E2 803E[7700]00                   cmp     byte [fhave96], 0     ; Do we have changeline       support?
000005E7 740A                           jz      short mChk_NoChangeLine       ; Brif not
000005E9 E83615                         call    mediacheck      ;  Call into removable routine
000005EC 7236                           jb      short err_exitj
```

```
13766 000005EE E89A16                     call    haschange
13767 000005F1 7512                       jnz     short Media_Done
13768                                 mChk_NoChangeLine:
13769                                     ; If we come here, we have a floppy with no changeline support
13770
13771 000005F3 BE0100                     mov     si, 1           ; Presume no change
13772 000005F6 A0[1E01]                   mov     al, [tim_drv]   ; Last drive accessed
13773 000005F9 263A4504                   cmp     al, [es:di+4]   ; [es:di+BDS.drivenum]
13774                                                             ; Is drive of last access the same?
13775 000005FD 7505                       jnz     short Media_Unk       ; No, then "i don't know"
13776 000005FF E82800                     call    Check_Time_Of_Access
13777 00000602 EB01                       jmp     short Media_Done
13778                                 ; ---------------------------------------------------------------------------
13779
13780                                 Media_Unk:
13781 00000604 4E                         dec     si              ; 0 ; Return "I don't know"
13782
13783                                     ; SI now contains the correct value for media change.
13784                                     ; Clean up the left overs
13785                                 Media_Done:
13786                                     ; 19/10/2022
13787 00000605 06                         push    es
13788 00000606 C41E[1200]                 les     bx, [ptrsav]
13789 0000060A 2689770E                   mov     [es:bx+0Eh], si       ; [es:bx+trans]
13790 0000060E 07                         pop     es
13791 0000060F 09F6                       or      si, si
13792 00000611 790F                       jns     short ret_carry_clear ;       volidok
13793 00000613 803E[7700]00               cmp     byte [fhave96], 0
13794 00000618 7403                       jz      short mChk1_NoChangeLine ; Brif      no changeline support
13795 0000061A E80016                     call    media_set_vid
13796                                 mChk1_NoChangeLine:
13797 0000061D C606[1E01]FF               mov     byte [tim_drv], 0FFh ; -1
13798                                                             ; Make sure we ask rom for media check
13799                                 ret_carry_clear:
13800 00000622 F8                         clc                     ; volidok
13801 00000623 C3                         retn
13802                                 ; ---------------------------------------------------------------------------
13803
13804                                 err_exitj:
13805 00000624 E88207                     call    maperror        ; guaranteed to       set carry
13806                                 ret81:
13807 00000627 B481                       mov     ah, 81h         ; return error status
13808 00000629 C3                         retn                    ; return with carry set
13809
13810                                 ; =============== S U B   R O U T I N E =======================================
13811
13812                                 ; ---------------------------------------------------------------------------
13813                                 ; perform a check on the time passed since the last access for this physical
13814                                 ; drive.
13815                                 ; we are accessing the same drive. if the time of last successful access was
13816                                 ; less than 2 seconds ago, then we may presume that the disk was not changed.
13817                                 ; returns in si:
13818                                 ;   0 - if time of last access was >= 2 seconds
13819                                 ;   1 - if time was < 2 seconds (i.e no media change assumed)
13820                                 ; registers affected ax,cx,dx, flags.
13821                                 ;
13822                                 ;   assume es:di -> bds, ds->Bios_Data
13823                                 ; ---------------------------------------------------------------------------
13824
13825                                     ; 21/12/2023 - Retro DOS v5.0 IBMBIO.COM
13826                                     ; 19/10/2022
13827                                 Check_Time_Of_Access:
13828 0000062A BE0100                     mov     si, 1           ; presume no change.
13829 0000062D E801FF                     call    GetTickCnt      ; cx:dx is the elapsed time
13830                                     ; 21/12/2023
13831 00000630 268B4579                   mov     ax, [es:di+79h]
13832                                     ;mov    ax, [es:di+47h]       ; [es:di+BDS.tim_lo]
13833                                                             ; get stored time
13834 00000634 29C2                       sub     dx, ax
13835                                     ; 21/12/2023
13836 00000636 268B457B                   mov     ax, [es:di+7Bh]
13837                                     ;mov    ax, [es:di+49h]       ; [es:di+BDS.tim_hi]
13838 0000063A 19C1                       sbb     cx, ax
13839                                     ; 11/08/2023
13840                                     ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0646h
13841                                     ;mov    al, [accesscount]
13842 0000063C 7515                       jnz     short timecheck_unk ; cx<>0 => >1 hour
13843 0000063E 09D2                       or      dx, dx          ; time must pass
13844 00000640 750C                       jnz     short timepassed ; yes, examine max value
13845                                     ; 11/08/2023
13846                                     ;inc    al
13847                                     ;cmp    al, 5
13848                                     ;;inc   byte [accesscount]
13849                                     ;;cmp   byte [accesscount], 5
13850                                                             ; if count is less than threshold, ok
13851                                     ;jb     short timecheck_ret
13852                                     ;;dec   byte [accesscount] ; don't let the count wrap
13853                                     ; 11/08/2023
13854                                     ;dec    al
13855                                     ;jmp    short timecheck_unk ; "i don't know" if media changed
13856                                     ; 11/08/2023
13857 00000642 803E[1D01]04               cmp     byte [accesscount], 4
13858 00000647 730A                       jnb     short timecheck_unk
13859 00000649 FE06[1D01]                 inc     byte [accesscount]
13860 0000064D C3                         retn
13861
13862                                 ; ---------------------------------------------------------------------------
13863
13864                                 timepassed:
13865 0000064E 83FA24                     cmp     dx, 36          ; 18*2 ; 18.2 tics per second.
13866                                                             ; min elapsed time? (2 seconds)
13867 00000651 7601                       jbe     short timecheck_ret ; yes, presume no change
13868
13869                                     ; everything indicates that we do not know what has happened.
13870                                 timecheck_unk:
13871 00000653 4E                         dec     si              ; presume i don't know
13872                                 timecheck_ret:
13873                                     ; 11/08/2023
13874                                     ;mov    [accesscount], al
13875 00000654 C3                         retn
13876
13877                                 ; ---------------------------------------------------------------------------
13878                                 ; 15/10/2022
13879                                 Err_Exitj2:
13880 00000655 EBCD                       jmp     short err_exitj
13881
13882                                 ; ---------------------------------------------------------------------------
13883
13884                                 ; 15/10/2022
13885
13886                                 ; ===========================================================================
13887                                 ; Build a valid bpb for the disk in the drive.
13888                                 ; ===========================================================================
13889
```

```
                                        ; 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
```

```
14014                                   ; mov_media_ids.. see if we can save some space by
14015                                   ; merging them... jgl
14016
14017                                   ;mov    cx, 11          ; size_of_EXT_BOOT_VOL_LABEL
14018                                   ; 10/12/2022
14019 000006AE B10B                     mov     cl, 11 ; cx = 11
14020
14021                                   ;;mov   si, offset vol_no_name ; "NO NAME    "
14022                                   ;mov    si, vol_no_name        ; 19/10/2022
14023                                   ; 22/12/2023
14024                                   ;mov    si, offset nul_vid ; "NO NAME    "
14025 000006B0 BE[6305]                 mov     si, nul_vid
14026
14027                                   ; 21/12/2023
14028 000006B3 83C77D                   add     di, 125
14029                                   ;add    di, 75          ; BDS.volid
14030
14031                                   ;rep movsb
14032                                   ; 21/12/2023
14033                                   ;rep movs byte ptr es:[di], byte ptr cs:[si] ; cs rep movsb
14034                                   ; 26/12/2023
14035                                   ;cs     ; vol_no_name is in BIOSCODE segment
14036                                   ;rep movsb
14037 000006B6 F3                       rep
14038 000006B7 2E                       cs
14039 000006B8 A4                       movsb
14040
14041                                   ; 11/09/2023 (BugFix, DI is not start addr of BDS structure here)
14042                                   ;;test byte [es:di+BDS.fatsiz], fbig
14043                                   ; (MSDOS 5.0 IO.SYS - BIOSCODE:05EFh)
14044                                   ;test   byte [es:di+1Fh], 40h
14045                                   ; 21/12/2023 - Retro DOS v5.0
14046                                   ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:06C3h)
14047                                   ;test   byte [es:di+59], 20h
14048                                          ; (here, es:di points to the BDS offset +136)
14049                                          ; purpose: test byte [es:di+BDS.fatsiz], fbigbig
14050                                          ; applied: test byte [es:BDS.fatsiz+136], fbigbig -BUG!-
14051
14052                                          ; (PCDOS 7.1 BUG note: 26/06/2023 - Erdogan Tan)
14053                                          ;; ! NOTE - 11/08/2023 - Erdogan Tan (Retro DOS v4.2 IO.SYS bugfix)
14054                                          ; Microsoft/IBM code has a bug here because the BDS's
14055                                          ; .volid and .filesys_id fields will be reset
14056                                          ; (to their default text) according to 'BDS.fatsiz' flags
14057                                          ; at the BDS offset 59 but current (this) code checks flags
14058                                          ; at ES:DI+59 while DI points the BDS offset 136!? ; (PCDOS 7.1)
14059                                          ;; at the BDS offset 31 but current (this) code checks flags
14060                                          ;; at ES:DI+31 while DI points the BDS offset 86!? ; (MSDOS 6.22)
14061                                          ;
14062                                          ; Correct Code:
14063                                          ; ;test byte [ES:59],20h or [ES:BDS.fatsiz],fbigbig  ; (PCDOS 7.1)
14064                                          ; ;;test byte [ES:31],40h or [ES:BDS.fatsiz],fbig  ; (MSDOS 6.22)
14065                                          ; 11/09/2023
14066                                          ; (before 'rep movsb') 'mov al,[es:di+BDS.Fatsiz]' and then
14067                                          ; (after 'rep movsb') 'test al,fbig' (AL is free/proper to use here)
14068                                          ;
14069                                          ; Same BUG is existing in MSDOS 6.22 IO.SYS - BIOSCODE:05EFh
14070                                          ; and in Windows ME IO.SYS - BIOSCODE:0E1Ah as 'test byte [es:di+59],20h'
14071
14072                                          ;
14073                                          ; (Why this bug did not affect MSDOS and PCDOS 7.x applications:
14074                                          ; 'clear_ids' is used for floppy disks only and the default
14075                                          ; option of 'clear_ids' is FAT12 volid and filesys_id text
14076                                          ; when the flag bit has wrong value for FAT16/40h or FAT32/20h.)
14077
14078                                   ; 21/12/2023 - Retro DOS v5.0
14079                                   ;mov    si, offset fat_32_id ; "FAT32   "
14080 000006B9 BE[5B05]                 mov     si, fat_32_id
14081
14082                                   ; 21/12/2023
14083                                   ; BugFix (of the PCDOS 7.1 IBMBIO.COM BUG) ; *+
14084                                   ;test   bl, fbigbig ; FAT32 flag
14085 000006BC F6C320                   test    bl, 20h ; * ; BL = [es:BDS.fatsiz] = [es:59]
14086 000006BF 750B                     jnz     short ci_bigfat
14087
14088                                   ;mov    si, offset fat_16_id ; "FAT16 "
14089 000006C1 BE[5305]                 mov     si, fat_16_id  ; 19/10/2022
14090
14091                                   ; 21/12/2023
14092                                   ; !BUG! (PCDOS 7.1 IBMBIO.COM BIOSCODE:06CDh)
14093                                   ;test   byte [es:di+59], 40h ; [es:di+BDS.fatsiz], fbig
14094                                   ; BugFix ; *+
14095                                   ;test   bl, fbig ; FAT16 flag
14096 000006C4 F6C340                   test    bl, 40h ; * ; Retro DOS v5.0
14097                                   ;;test  al, 40h ; * ; Retro DOS v4.2
14098 000006C7 7503                     jnz     short ci_bigfat
14099
14100                                   ;mov    si, offset fat_12_id ; "FAT12 "
14101 000006C9 BE[4B05]                 mov     si, fat_12_id  ; 19/10/2022
14102                                 ci_bigfat:
14103                                           ;mov    cx, 8           ; size_of_EXT_SYSTEM_ID
14104                                           ; 10/12/2022
14105 000006CC B108                             mov     cl, 8 ; cx = 8
14106 000006CE 83C705                           add     di, 5           ; (BDS.filesys_id-BDS.volid)-size_of_EXT_BOOT_VOL_LABEL
14107                                                                   ; filesys_id field
14108                                           ;rep movsb
14109                                           ; 21/12/2023 - Retro DOS v5.0
14110                                           ;rep movs byte ptr es:[di], byte ptr cs:[si] ; 0F3h,2Eh,0A4h
14111                                           ; 26/12/2023
14112                                           ;cs     ; fat32_id, fat16_id and fat12_id are in BIOSCODE segment
14113                                           ;rep movsb
14114 000006D1 F3                               rep
14115 000006D2 2E                               cs
14116 000006D3 A4                               movsb
14117
14118 000006D4 5F                               pop     di              ; restore bds pointer
14119                                 getret_exit:                    ; 21/12/2023
14120 000006D5 C3                               retn
14121
14122                                 ; =============== S U B   R O U TI N E =======================================
14123
14124                                 ; 15/10/2022
14125
14126                                 ; ---------------------------------------------------------------------------
14127                                 ;   getbp - return bpb from the drive specified by the bds.
14128                                 ;       if the return_fake_bpb flag is set, then it does nothing.
14129                                 ;       note that we never come here for fixed disks.
14130                                 ;       for all other cases,
14131                                 ;         - it reads boot sector to pull out the bpb
14132                                 ;         - if no valid bpb is found, it then reads the fat sector,
14133                                 ;           to get the fat id byte to build the bpb from there.
14134                                 ;
14135                                 ;   inputs: es:di point to correct bds.
14136                                 ;
14137                                 ;   outputs:       fills in bpb in current bds if valid bpb or fat id on disk.
```

```
14138                                   ;           carry set, and al=7 if invalid disk.
14139                                   ;           carry set and error code in al if other error.
14140                                   ;           if failed to recognize the boot record, then will set the
14141                                   ;           set_id_flag to 0.
14142                                   ;           this routine will only work for a floppy diskette.
14143                                   ;                for a fixed disk, it will just return.
14144                                   ;
14145                                   ;   ****** Note: getbp is a clone of getbp which uses the newer
14146                                   ;     segment definitions. It should be migrated towards.
14147                                   ;      now es:di has the bds, ds: has Bios_Data
14148                                   ; ---------------------------------------------------------------------------
14149
14150                                           ; 29/12/2023
14151                                           ; 21/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
14152                                   GetBp:
14153                                           ; if returning fake bpb then return bpb as is.
14154                                           ; 21/12/2023
14155 000006D6 26F6453F05                       test    byte [es:di+3Fh], 5 ; PCDOS 7.1
14156                                           ;test   byte [es:di+BDS.flags], return_fake_bpb|fnon_removable
14157                                           ;test   byte [es:di+23h], 5 ; MSDOS 6.22 (& MSDOS 5.0)
14158                                           ;jz     short getbp1   ; getbp1
14159                                           ;jmp    getret_exit
14160                                           ; 21/12/2023
14161 000006DB 75F8                            jnz     short getret_exit
14162                                   ; ---------------------------------------------------------------------------
14163                                   getbp1:
14164 000006DD 51                               push    cx
14165 000006DE 52                               push    dx
14166 000006DF 53                               push    bx
14167
14168                                           ; attempt to read in boot sector and determine bpb.
14169                                           ; we assume that the 2.x and greater dos disks all
14170                                           ; have a valid boot sector.
14171
14172 000006E0 E8CF00                           call    readbootsec
14173 000006E3 720A                             jb      short getbp_err_ret_brdg ; carry set if there was error.
14174 000006E5 09DB                             or      bx, bx          ; bx is 0 if boot sector is valid.
14175 000006E7 7509                             jnz     short dofatbpb
14176 000006E9 E81401                           call    movbpb          ; move bpb into       registers
14177                                           ;jmp    short Has1
14178                                           ; 21/12/2023 - Retro DOS v5.0 (PCDOS 7.1) IBMBIO.COM
14179 000006EC E9B500                           jmp     getret
14180                                   ; ---------------------------------------------------------------------------
14181
14182                                   getbp_err_ret_brdg:
14183 000006EF E9B600                           jmp     getbp_err_ret
14184                                   ; ---------------------------------------------------------------------------
14185
14186                                           ; we have a 1.x diskette. In this case read in the fat ID byte
14187                                           ; and fill in bpb from there.
14188                                   dofatbpb:
14189 000006F2 E8B401                           call    readfat         ; puts media descriptor       byte in ah
14190 000006F5 72F8                             jb      short getbp_err_ret_brdg
14191                                           ;cmp    ds:fhave96, 0  ;  changeline support available?
14192 000006F7 803E[7700]00                     cmp     byte [fhave96], 0 ; 19/10/2022
14193 000006FC 7403                             jz      short bpb_nochangeline ; brif not
14194 000006FE E83115                           call    hidensity       ; may not return! May add sp, 2       and
14195                                                                   ; jump to has1!!!!!! or       has720K
14196                                   bpb_nochangeline:         ; test for a valid 3.5" medium
14197                                           ; 21/12/2023 - Retro DOS v5.0
14198 00000701 26807D3E02                       cmp     byte [es:di+3Eh], 2
14199                                           ;cmp    byte [es:di+22h], 2 ; [es:di+BDS.formfactor]
14200                                                                   ; ffSmall
14201 00000706 7512                             jnz     short is_floppy
14202 00000708 80FCF9                           cmp     ah, 0F9h        ; is it a valid fat id byte for       3.5" ?
14203 0000070B 7512                             jnz     short got_unknown_medium
14204                                   Has720K:
14205                                           ; 21/12/2023
14206                                           ;;mov   bx, offset sm92 ; pointer to correct bpb
14207                                           ;mov    bx, sm92        ; 19/10/2022
14208
14209                                           ; es points to segment of bds. the following should be modified
14210                                           ; to get spf,csec,spau,spt correctly. it had been wrong if
14211                                           ; driver.sys is loaded since the bds is inside the driver.sys.
14212
14213                                           ; 21/12/2023
14214                                           ;; 10/12/2022
14215                                           ;;mov   al, [bx+0]      ; [bx+bpbtype.spf]
14216                                           ;; 21/12/2022
14217                                           ;mov    al, [bx]
14218                                           ;mov    cx, [bx+3]      ; [bx+bpbtype.csec]
14219                                           ;mov    dx, [bx+5]      ; [bx+bpbtype.spau]
14220                                           ;mov    bx, [bx+1]      ; [bx+bpbtype.spt]
14221                                           ;; 19/10/2022 - Temporary !
14222                                           ;;db    8Ah, 87h, 0, 0 ; mov al, [bx+0]
14223                                           ;;db    8Bh, 8Fh, 3, 0 ; mov cx, [bx+3]
14224                                           ;;db    8Bh, 97h, 5, 0 ; mov dx, [bx+5]
14225                                           ;;db    8Bh, 9Fh, 1, 0 ; mov bx, [bx+1]
14226
14227                                           ; 21/12/2023 - Retro DOS v5.0
14228 0000070D B003                            mov     al, 3           ; bpbtype.sbf = 3
14229 0000070F B9A005                          mov     cx, 1440        ; bpbtype.csec = 1440
14230 00000712 BA0202                          mov     dx, 202h        ; dl = bpbtype.spau = 2
14231                                                                   ; dh = bpbtype.chead = 2
14232 00000715 BB0970                          mov     bx, 7009h       ; bl = bpbtype.spt = 9
14233                                                                   ; bh = bpbtype.dire = 112
14234 00000718 EB30                            jmp     short Has1
14235                                   ; ---------------------------------------------------------------------------
14236
14237                                   is_floppy:                ; must be a 5.25" floppy if we come here
14238 0000071A 80FCF8                          cmp     ah, 0F8h        ; valid media?? (0F8h-0FFh)
14239                                           ;jb     short got_unknown_medium
14240                                           ; 21/12/2023
14241 0000071D 730A                            jnb     short chk_160K
14242                                   ; ---------------------------------------------------------------------------
14243                                           ; 21/12/2023
14244                                           ; we have a 3.5" diskette for which we cannot build a bpb.
14245                                           ; we do not assume any type of bpb for this medium.
14246                                   got_unknown_medium:
14247                                           ;mov    ds:set_id_flag,       0
14248 0000071F C606[9B04]00                     mov     byte [set_id_flag], 0
14249 00000724 B007                            mov     al, 7
14250 00000726 F9                              stc
14251 00000727 EB7B                            jmp     short getret
14252                                   ; ---------------------------------------------------------------------------
14253                                   chk_160K:
14254 00000729 B001                            mov     al, 1           ; set number of       fat sectors
14255 0000072B BB0840                          mov     bx, 16392       ; 64*256+8
14256                                                                   ; set dir entries and sector max
14257 0000072E B94001                          mov     cx, 320         ; 40*8
14258                                                                   ; set size of drive
14259 00000731 BA0101                          mov     dx, 257         ; 01*256+1
14260                                                                   ; set head limit and sec/all unit
14261                                           ; 21/12/2023
```

```
14262                                   ;mov    al, 1           ; bpbtype.sbf = 1
14263                                   ;mov    bx, 4008h       ; bl = bpbtype.spt = 8
14264                                   ;                       ; bh = bpbtype.dire = 64
14265                                   ;mov    cx, 140h        ; bpbtype.csec = 320
14266                                   ;mov    dx, 101h        ; dl = bpbtype.spau = 1
14267                                   ;                       ; dh = bpbtype.chead = 1
14268
14269 00000734 F6C402                   test    ah, 2           ; test for 8 or       9 sector
14270 00000737 7505                     jnz     short has8      ; nz = has 8 sectors
14271
14272                                   ; 29/12/2023
14273                                   ;inc    al      ; 2     ; inc number of       fat sectors
14274                                   ;inc    bl      ; 9     ; inc sector max
14275 00000739 40                       inc     ax
14276 0000073A 43                       inc     bx
14277
14278                                   ;add    cx, 40          ; increase size       (to 360)
14279                                   ; 18/12/2022
14280 0000073B 80C128                   add     cl, 40  ; 28h   ; 180K (360 sectors)
14281                           has8:
14282 0000073E F6C401                   test    ah, 1           ; test for 1 or       2 heads
14283 00000741 7407                     jz      short Has1      ; jz = 1 head
14284 00000743 01C9                     add     cx, cx          ; double size of disk
14285 00000745 B770                     mov     bh, 112         ; increase number of directory entries
14286 00000747 FEC6                     inc     dh      ; 2     ; inc sec/all unit
14287                                   ; 29/12/2023
14288                                   ;inc    dl      ; 2     ; inc head limit
14289 00000749 42                       inc     dx
14290                           Has1:
14291                                   ; 02/09/2023 (PCDOS 7.1, IBMBIO.COM - BIOSCODE:0754h)
14292 0000074A 1E                       push    ds
14293 0000074B 06                       push    es
14294 0000074C 1F                       pop     ds
14295
14296                                   ;mov    [es:di+8], dh  ; [es:di+BDS.secperclus]
14297                                   ;mov    [es:di+0Ch], bh        ; [es:di+BDS.direntries]
14298                                   ;mov    [es:di+0Eh], cx        ; [es:di+BDS.totalsecs16]
14299                                   ;mov    [es:di+10h], ah        ; [es:di+BDS.media]
14300                                   ;mov    [es:di+11h], al        ; [es:di+BDS.fatsecs]
14301                                   ;mov    [es:di+13h], bl        ; [es:di+BDS.secpertrack]
14302                                   ;mov    [es:di+15h], dl        ; [es:di+BDS.heads]
14303
14304 0000074D 887508                   mov     [di+8], dh      ; [di+BDS.secperclus]
14305 00000750 30F6                     xor     dh, dh
14306 00000752 895515                   mov     [di+15h], dx    ; [di+BDS.heads]
14307 00000755 88FA                     mov     dl, bh
14308 00000757 89550C                   mov     [di+0Ch], dx    ; [di+BDS.direntries]
14309 0000075A 894D0E                   mov     [di+0Eh], cx    ; [di+BDS.totalsecs16]
14310 0000075D 894D1B                   mov     [di+1Bh], cx    ; [di+BDS.totalsecs32]
14311 00000760 886510                   mov     [di+10h], ah    ; [di+BDS.media]
14312 00000763 88C2                     mov     dl, al
14313 00000765 895511                   mov     [di+11h], dx    ; [di+BDS.fatsecs]
14314 00000768 88DA                     mov     dl, bl
14315 0000076A 895513                   mov     [di+13h], dx    ; [di+BDS.secpertrack]
14316
14317                                   ; the BDS_BPB.BPB_HIDDENSECTORS+2 field and the
14318                                   ; BDS_BPB.BPB_BIGTOTALSECTORS field need to be set
14319                                   ; to 0 since this code is for floppies
14320
14321                                   ; 18/12/2022
14322                                   ;mov    word [es:di+19h], 0 ; [es:di+BDS.hiddensecs+2]
14323                                   ;mov    word [es:di+17h], 0 ; [es:di+BDS.hiddensecs]
14324                                   ;mov    word [es:di+1Dh], 0 ; [es:di+BDS.totalsecs32+2]
14325                                   ; 18/12/2022
14326 0000076D 29C9                     sub     cx, cx ; 0
14327                                   ;mov    [es:di+19h], cx ; 0 ; [es:di+BDS.hiddensecs+2]
14328                                   ;mov    [es:di+17h], cx ; 0 ; [es:di+BDS.hiddensecs]
14329                                   ;mov    [es:di+1Dh], cx ; 0 ; [es:di+BDS.totalsecs32+2]
14330
14331                                   ; 02/09/2023
14332 0000076F 894D19                   mov     [di+19h], cx ; 0 ; [di+BDS.hiddensecs+2]
14333 00000772 894D17                   mov     [di+17h], cx ; 0 ; [di+BDS.hiddensecs]
14334 00000775 894D1D                   mov     [di+1Dh], cx ; 0 ; [di+BDS.totalsecs32+2]
14335
14336                                   ; 21/12/2023 - Retro DOS v5.0
14337 00000778 894D1F                   mov     [di+1Fh], cx    ; [di+BDS.fatsecs32] ; BPB_FATSz32
14338 0000077B 894D21                   mov     [di+21h], cx    ; [di+BDS.fatsecs32+2]
14339 0000077E 894D27                   mov     [di+27h], cx    ; [di+BDS.rootdirclust]
14340 00000781 894D29                   mov     [di+29h], cx    ; [di+BDS.rootdirclust+2]
14341 00000784 894D2F                   mov     [di+2Fh], cx    ; [di+BDS.reserved]
14342                                                           ;     BPB_Reserved (12 zero bytes)
14343 00000787 894D31                   mov     [di+31h], cx
14344 0000078A 894D33                   mov     [di+33h], cx
14345 0000078D 894D35                   mov     [di+35h], cx
14346 00000790 894D37                   mov     [di+37h], cx
14347 00000793 894D39                   mov     [di+39h], cx
14348 00000796 894D23                   mov     [di+23h], cx    ; [di+BDS.extflags] ; BPB_ExtFlags
14349 00000799 894D25                   mov     [di+25h], cx    ; [di+BDS.fsver] ; BPB_FSVer
14350
14351 0000079C 49                       dec     cx              ; -1 ; 0FFFFFFFFh
14352 0000079D 894D2B                   mov     [di+2Bh], cx    ; [di+BDS.fsinfo] ; BPB_FSInfo
14353 000007A0 894D2D                   mov     [di+2Dh], cx    ; [di+BDS.bkbootsec] ; BPB_BkBootSec
14354
14355 000007A3 1F                       pop     ds ; 02/09/2023
14356                           getret:
14357 000007A4 5B                       pop     bx
14358 000007A5 5A                       pop     dx
14359 000007A6 59                       pop     cx
14360                           ;getret_exit:           ; 21/12/2023
14361 000007A7 C3                       retn
14362                           ; ---------------------------------------------------------------------------
14363
14364                           getbp_err_ret:  ; before doing anything else, set set_id_flag to 0.
14365                                   ;mov    ds:set_id_flag,       0
14366                                   ; 19/10/2022
14367 000007A8 C606[9B04]00             mov     byte [set_id_flag], 0
14368 000007AD E8F905                   call    maperror
14369 000007B0 EBF2                     jmp     short getret
14370                           ; ---------------------------------------------------------------------------
14371                           ; 21/12/2023
14372                                   ;       ; we have a 3.5" diskette for which we cannot build a bpb.
14373                                   ;       ; we do not assume any type of bpb for this medium.
14374                                   ;
14375                           ;got_unknown_medium:
14376                                   ;       ;mov    ds:set_id_flag,       0
14377                                   ;       mov     byte [set_id_flag], 0
14378                                   ;       mov     al, 7
14379                                   ;       stc
14380                                   ;       jmp     short getret
14381
14382                           ; =============== S U B   R O U T I N E =======================================
14383
14384                           ; 15/10/2022 - Retro DOS 4.0 (Modified MSDOS 5.0 IO.SYS)
14385
```

```
14386                                   ; ----------------------------------------------------------------
14387                                   ; read in the boot sector. set carry if error in reading sector.
14388                                   ; bx is set to 1 if the boot sector is invalid, otherwise it is 0.
14389                                   ;
14390                                   ;   assumes es:di -> bds, ds-> Bios_Data
14391                                   ; ----------------------------------------------------------------
14392
14393                                   ; 10/03/2019 - Retro DOS v4.0
14394
14395                                   ; 30/12/2022 - Retro DOS v4.2
14396                                   ; (MSDOS 6.21 IO.SYS, BIOSCODE:06C3h)
14397                                   ; ((MSDOS 6.22 IO.SYS, BIOSCODE:06C3h)) ; 22/12/2023
14398
14399                                   ; 22/12/2023 - Retro DOS v5.0
14400                                   ; (PCDOS 7.1 IBMBIO.COM, BIOSCODE:07C6h)
14401
14402                                   readbootsec:
14403 000007B2 B600                               mov     dh, 0           ; head 0
14404 000007B4 B90100                             mov     cx, 1           ; cylinder 0, sector 1
14405 000007B7 E8FC00                             call    read_sector
14406 000007BA 7243                               jb      short err_ret
14407 000007BC 31DB                               xor     bx, bx          ; assume valid boot sector
14408
14409                                             ; put a sanity check for the boot sector in here to detect
14410                                             ; boot sectors that do not have valid bpbs. we examine the
14411                                             ; first two bytes - they must contain a long jump (69h) or a
14412                                             ; short jump (EBh) followed by a nop (90h), or a short jump
14413                                             ; (E9h). if this test is passed, we further check by examining
14414                                             ; the signature at the end of the boot sector for the word
14415                                             ; AA55h. if the signature is not present, we examine the media
14416                                             ; descriptor byte to see if it is valid. for dos 3.3, this
14417                                             ; logic is modified a little bit. we are not going to check
14418                                             ; signature. instead we are going to sanity check the media
14419                                             ; byte in bpb regardless of the validity of signature. this is
14420                                             ; to save the already developed commercial products that have
14421                                             ; good jump instruction and signature but with the false bpb
14422                                             ; informations
14423
14424                                   ; that will crash the diskette drive operation. (for example, symphony diskette).
14425
14426                                             ; 02/09/2023
14427                                             ; 19/10/2022
14428                                             ;cmp    byte [disksector], 69h ; is it a direct jump?
14429                                             ;jz     short check_bpb_mediabyte ; don't need to find a nop
14430                                             ;cmp    byte [disksector], 0E9h ; dos 2.0 jump?
14431                                             ;jz     short check_bpb_mediabyte ; no need for     nop
14432                                             ;cmp    byte [disksector], 0EBh ; how about a short jump?
14433                                             ;jnz    short invalidbootsec
14434                                             ;cmp    byte [disksector+2], 90h ; is next one a nop?
14435                                             ;jnz    short invalidbootsec
14436
14437                                             ; 02/09/2023 (PCDOS 7.1)
14438 000007BE A0[5201]                           mov     al, [disksector]
14439 000007C1 3C69                               cmp     al, 69h         ; is it a direct jump?
14440 000007C3 740F                               je      short check_bpb_mediabyte
14441                                                                     ; don't need to find a nop
14442 000007C5 3CE9                               cmp     al, 0E9h        ; dos 2.0 jump?
14443 000007C7 740B                               je      short check_bpb_mediabyte
14444                                                                     ; no need for nop
14445 000007C9 3CEB                               cmp     al, 0EBh        ; how about a short jump?
14446 000007CB 7530                               jne     short invalidbootsec
14447 000007CD 803E[5401]90                       cmp     byte [disksector+2], 90h ; is next one a nop?
14448 000007D2 7529                               jne     short invalidbootsec
14449
14450                                   ; 15/10/5022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
14451                                   ;
14452                                   ;; 10/03/2019
14453                                   ;; (MSDOS 3.3, MSDISK.ASM, 1988)
14454                                   ;;
14455                                   ;; Don't have to perform the following signature check since
14456                                   ;; we need to check the media byte even with the good signatured diskette.
14457                                   ;;
14458                                   ;;check_signature:
14459                                   ;;          cmp     word [cs:disksector+1FEh],0AA55h ; see if non-ibm
14460                                   ;;                                                   ; disk or 1.x media.
14461                                   ;;          jz      short checksinglesided ; go see if singled sided medium.
14462                                   ;;                                         ; may need some special handling
14463
14464                                   ; check for non-ibm disks which do not have the signature AA55h at the
14465                                   ; end of the boot sector, but still have a valid boot sector. this is done
14466                                   ; by examining the media descriptor in the boot sector.
14467
14468                                             ; 19/10/2022
14469                                   check_bpb_mediabyte:
14470 000007D4 A0[6701]                           mov     al, [disksector+15h]
14471                                                                     ; [disksector+EXT_BOOT.BPB+EBPB.MEDIADESCRIPTOR]
14472 000007D7 50                                 push    ax ; 02/09/2023
14473 000007D8 24F0                               and     al, 0F0h
14474 000007DA 3CF0                               cmp     al, 0F0h        ; allow for strange media
14475 000007DC 58                                 pop     ax ; 02/09/2023
14476 000007DD 751E                               jnz     short invalidbootsec
14477
14478                                   ; there were some (apparently a lot of them) diskettes that had been formatted
14479                                   ; under dos 3.1 and earlier versions which have invalid bpbs in their boot
14480                                   ; sectors. these are specifically diskettes that were formatted in drives
14481                                   ; with one head, or whose side 0 was bad. these contain bpbs in the boot
14482                                   ; sector that have the sec/clus field set to 2 instead of 1, as is standard
14483                                   ; in dos. in order to support them, we have to introduce a "hack" that will
14484                                   ; help our build bpb routine to recognise these specific cases, and to
14485                                   ; set up out copy of the bpb accordingly.
14486                                   ; we do this by checking to see if the boot sector is off a diskette that
14487                                   ; is single-sided and is a pre-dos 3.20 diskette. if it is, we set the
14488                                   ; sec/clus field to 1. if not, we carry on as normal.
14489
14490                                   checksinglesided:
14491                                             ;mov    al, [disksector+15h]
14492                                             ; 02/09/2023
14493                                             ; al = [disksector+15h]
14494 000007DF 3CF0                               cmp     al, 0F0h
14495 000007E1 741B                               jz      short gooddsk
14496 000007E3 A801                               test    al, 1
14497 000007E5 7517                               jnz     short gooddsk
14498 000007E7 813E[5A01]332E                     cmp     word [disksector+8], 2E33h ; "3."
14499 000007ED 7507                               jnz     short mustbeearlier
14500 000007EF 803E[5C01]32                       cmp     byte [disksector+0Ah], 32h ; "2"
14501 000007F4 7308                               jnb     short gooddsk
14502
14503                                   ; we must have a pre-3.20 diskette. set the sec/clus field to 1
14504
14505                                   mustbeearlier:
14506 000007F6 C606[5F01]01                       mov     byte [disksector+0Dh], 1
14507                                                                   ; [disksector+EXT_BOOT.BPB+EBPB.SECTORSPERCLUSTER]
14508 000007FB EB01                               jmp     short gooddsk
14509                                   ; ----------------------------------------------------------------------------
```

```
14510
14511                                   invalidbootsec:
14512 000007FD 43                               inc     bx              ; indicate that boot sector invalid
14513                                           ; 10/12/2022
14514                                   movbpb_ret:
14515                                   gooddsk:
14516 000007FE F8                               clc
14517                                   err_ret:
14518 000007FF C3                               retn
14519                                   ; ---------------------------------------------------------------------------
14520
14521                                           ; 10/12/2022
14522                                   ;err_ret:
14523                                           ;retn
14524
14525                                   ; =============== S U B   R O U T I N E =======================================
14526
14527                                   ; 15/10/2022
14528                                   ; ---------------------------------------------------------------------------
14529                                   ; 'movbpb' moves the bpb read from the boot sector into registers for use by
14530                                   ; getbp routine at has1
14531                                   ;
14532                                   ; if the set_id_flag is 1, and if an extended boot record, then set volume
14533                                   ; serial number, volume label, file system id in bds according to
14534                                   ; the boot record. after that, this routine will set the set_id_flag to 2
14535                                   ; to signal that volume label is set already from the extended boot record
14536                                   ; (so, don't set it again by calling "set_volume_id" routine which uses
14537                                   ; the volume label in the root directory.)
14538                                   ; ---------------------------------------------------------------------------
14539
14540                                   ; 10/03/2019 - Retro DOS v4.0
14541
14542                                   ; 22/12/2023
14543                                   %if 0
14544                                           ; 19/10/2022
14545                                   movbpb:
14546                                           mov     dh, [disksector+0Dh]
14547                                                   ; disksector+EXT_BOOT.BPB+EBPB.SECTORSPERCLUSTER]
14548                                                   ; sectors per unit
14549                                           mov     bh, [disksector+11h]
14550                                                   ; [disksector+EXT_BOOT.BPB+EBPB.ROOTENTRIES]
14551                                                   ; number of directory entries
14552                                           mov     cx, [disksector+13h]
14553                                                   ; [disksector+EXT_BOOT.BPB+EBPB.TOTALSECTORS]
14554                                                   ; size of drive
14555                                           mov     ah, [disksector+15h]
14556                                                   ; [disksector+EXT_BOOT.BPB+EBPB.MEDIADESCRIPTOR]
14557                                                   ; media descriptor
14558                                           mov     al, [disksector+16h];
14559                                                   ; [disksector+EXT_BOOT.BPB+EBPB.SECTORSPERFAT]
14560                                                   ; number of fat       sectors
14561                                           mov     bl, [disksector+18h]
14562                                                   ; [disksector+EXT_BOOT.BPB+EBPB.SECTORSPERTRACK]
14563                                                   ; sectors per track
14564                                           mov     dl, [disksector+1Ah]
14565                                                   ; [disksector+EXT_BOOT.BPB+EBPB.HEADS]
14566                                                   ; number of heads
14567                                   %else
14568                                           ; 29/12/2023
14569                                           ; 22/12/2023 - Retro DOS v5.0
14570                                           ; (PCDOS 7.1 IBMBIO.COM, BIOSCODE:0814h)
14571                                           ;;;
14572                                   movbpb:
14573 00000800 57                               push    di
14574 00000801 83C706                           add     di, 6           ; BDS+6 = BDS.BPB
14575 00000804 8D36[5D01]                       lea     si, [disksector+0Bh]
14576 00000808 B93500                           mov     cx, 53          ; copy bios parameters block
14577                                                                   ; from BPB_BytsPerSec to (FAT32) BS_DrvNum (excluded)
14578 0000080B FC                               cld
14579 0000080C F3A4                             rep movsb
14580 0000080E 8B4CD3                           mov     cx, [si-45]     ; si = disksector+64 -> 64-45 = 19
14581                                                                   ; disksektor+19 = BPB_TotSec16
14582 00000811 31C0                             xor     ax, ax
14583 00000813 E308                             jcxz    movbpb_bigdisk
14584 00000815 26894DE0                         mov     [es:di-32], cx ; write 16 bit total sectors
14585                                                                   ; to 32 bit total sectors field
14586 00000819 268945E2                         mov     [es:di-30], ax ; BPB_TotalSec32+2 (BDS offset 29, BPB offset 23)
14587                                   movbpb_bigdisk:
14588 0000081D 3944D6                           cmp     [si-42], ax     ; BPB_FATSz16 = disksector+22
14589 00000820 7410                             jz      short movbpb_fat32
14590                                   movbpb_fat:
14591 00000822 83EF1C                           sub     di, 28          ; di = BDS offset 31 (BPB offset 25)
14592                                           ; 29/12/2023
14593 00000825 B90C00                           mov     cx, 12          ; clear 12 byte extended BDS (FAT32) fields
14594                                                                   ; (which are used only for FAT32 disks)
14595 00000828 F3AA                             rep stosb
14596 0000082A 48                               dec     ax              ; -1 ; 0FFFFh
14597 0000082B AB                               stosw                   ; set BDS offset 43 (dword) to -1
14598                                                                   ; dword [BDS.BPB_FSInfo] = 0FFFFFFFFh
14599 0000082C AB                               stosw
14600 0000082D 40                               inc     ax              ; ax = 0
14601 0000082E B10C                             mov     cl, 12
14602                                           ;mov    cx, 12          ; clear BDS offset 47 to 59
14603                                                                   ; (BPB offset 41 to 53) (disksector offset 52 to 64)
14604 00000830 F3AA                             rep stosb
14605                                   movbpb_fat32:
14606 00000832 5F                               pop     di
14607                                   %endif
14608                                           ;;;
14609
14610 00000833 803E[9B04]01                     cmp     byte [set_id_flag], 1 ; called by get_bpb?
14611 00000838 75C4                             jnz     short movbpb_ret
14612 0000083A E81200                           call    mov_media_ids
14613 0000083D 7205                             jb      short movbpb_conv ; conventional boot record?
14614 0000083F C606[9B04]02                     mov     byte [set_id_flag], 2 ; signals that volume id is set
14615                                   movbpb_conv:
14616 00000844 803E[7700]01                     cmp     byte [fhave96], 1
14617 00000849 75B3                             jnz     short movbpb_ret
14618 0000084B E83714                           call    resetchanged    ; reset flags in bds to       not fchanged.
14619                                           ; 10/12/2022
14620                                           ; cf = 0
14621                                   ;movbpb_ret:
14622                                           ;clc
14623 0000084E C3                               retn
14624
14625                                   ; =============== S U B   R O U T I N E =======================================
14626
14627                                   ;copy the boot_serial number, volume id, and filesystem id from the
14628                                   ;***extended boot record*** in ds:disksector to the bds table pointed
14629                                   ;by es:di.
14630
14631                                   ;in.) es:di -> bds
14632                                   ;     ds:disksector = valid extended boot record.
14633                                   ;out.) vol_serial, bds_volid and bds_system_id in bds are set according to
```

```
14634                                  ;      the boot record information.
14635                                  ;     carry flag set if not an extended bpb.
14636                                  ;     all registers saved except the flag.
14637                                  
14638                                  ; 22/12/2023
14639                                  %if 0
14640                                          ; 19/10/2022
14641                                  mov_media_ids:
14642                                          cmp     byte [disksector+26h], 29h
14643                                                                  ; [disksector+EXT_BOOT.SIG],
14644                                                                  ; EXT_BOOT_SIGNATURE
14645                                          jnz     short mmi_not_ext
14646                                          push    cx
14647                                          mov     cx, [disksector+27h]
14648                                                                  ; [disksector+EXT_BOOT.SERIAL]
14649                                          mov     [es:di+57h], cx         ; [es:di+BDS.vol_serial]
14650                                          mov     cx, [disksector+29h]
14651                                                                  ; [disksector+EXT_BOOT.SERIAL+2]
14652                                          mov     [es:di+59h], cx         ; [es:di+BDS.vol_serial+2]
14653                                          push    di
14654                                          push    si
14655                                          mov     cx, 11          ; size_of_EXT_BOOT_VOL_LABEL
14656                                          mov     si, disksector+2Bh
14657                                          ;mov    si, (offset disksector+2Bh) ;
14658                                                                  ; disksector+EXT_BOOT.VOL_LABEL
14659                                          add     di, 75          ; BDS.volid
14660                                          rep movsb
14661                                          ;mov    cx, 8           ; size_of_EXT_SYSTEM_ID
14662                                          ; 10/12/2022
14663                                          mov     cl, 8 ; cx = 8
14664                                          mov     si, disksector+36h
14665                                          ;mov    si, (offset disksector+36h) ; disksector+EXT_BOOT.SYSTEM_ID
14666                                          add     di, 5           ; (BDS.filesys_id-BDS.volid)-size_of_EXT_BOOT_VOL_LABEL
14667                                          rep movsb
14668                                          pop     si
14669                                          pop     di
14670                                          pop     cx
14671                                          ; 10/12/2022
14672                                          ; cf = 0
14673                                          ;clc            ; this clc is not required (16/06/2019 - Erdogan Tan)
14674                                                          ; (20/09/2022)
14675                                          retn
14676                                  %else
14677                                          ; 22/12/2023 - Retro DOS v5.0
14678                                          ; (PCDOS 7.1 IBMBIO.COM, BIOSCODE:0865h)
14679                                          ;;;
14680                                  mov_media_ids:
14681 0000084F 833E[6801]00                      cmp     word [disksector+16h], 0 ; BPB.FATSz16
14682 00000854 7507                              jnz     short mmi_chk_fat
14683 00000856 803E[9401]29                      cmp     byte [disksector+42h], 29h
14684                                                                    ; [disksector+FAT32_EXT_BOOT.SIG],
14685                                                                    ; EXT_BOOT_SIGNATURE
14686 0000085B EB05                              jmp     short mmi_chk_fat32
14687                                  mmi_chk_fat:
14688 0000085D 803E[7801]29                      cmp     byte [disksector+26h], 29h
14689                                                                    ; [disksector+EXT_BOOT.SIG],EXT_BOOT_SIGNATURE
14690                                  mmi_chk_fat32:
14691 00000862 7543                              jnz     short mmi_not_ext
14692 00000864 51                                push    cx
14693 00000865 50                                push    ax
14694 00000866 57                                push    di
14695 00000867 56                                push    si
14696 00000868 1E                                push    ds
14697 00000869 833E[6801]00                      cmp     word [disksector+16h], 0 ; BPB.FATSz16
14698 0000086E 750C                              jnz     short mmi_fat
14699
14700                                  mmi_fat32:                              ; FAT32 file system
14701                                          ;lds    cx, dword ptr ds:disksector+43h
14702 00000870 C50E[9501]                        lds     cx, [disksector+43h]  ; BS_FAT32_VolID
14703 00000874 BE[9901]                          mov     si, disksector+47h    ; BS_FAT32_VolLab
14704 00000877 B8[A401]                          mov     ax, disksector+52h    ; BS_FAT32_FilSysType
14705 0000087A EB0A                              jmp     short mmi_do
14706
14707                                  mmi_fat:
14708                                          ;lds    cx, dword ptr ds:disksector+27h
14709 0000087C C50E[7901]                        lds     cx, [disksector+27h]  ; BS_VolID
14710 00000880 BE[7D01]                          mov     si, disksector+2Bh    ; BS_VolLab
14711 00000883 B8[8801]                          mov     ax, disksector+36h    ; BS_FilSysType
14712                                  mmi_do:
14713 00000886 26898D8900                        mov     [es:di+89h], cx         ; [es:di+BDS.vol_serial]
14714                                                                    ; (BDS offset 137)
14715 0000088B 268C9D8B00                        mov     [es:di+8Bh], ds         ; [es:di+BDS.vol_serial+2]
14716 00000890 1F                                pop     ds
14717 00000891 B90B00                            mov     cx, 11
14718 00000894 83C77D                            add     di, 125         ; di = di+125 = BDS.volid
14719 00000897 F3A4                              rep movsb
14720 00000899 B108                              mov     cl, 8           ; di = di+136
14721 0000089B 89C6                              mov     si, ax          ; BS_FilSysType or BS_FAT32_FilSysType
14722 0000089D 83C705                            add     di, 5           ; di = di+141 = BDS.filesys_id
14723 000008A0 F3A4                              rep movsb
14724 000008A2 5E                                pop     si
14725 000008A3 5F                                pop     di
14726 000008A4 58                                pop     ax
14727 000008A5 59                                pop     cx
14728                                            ;clc    ; this clc is not required (16/06/2019 - Erdogan Tan)
14729                                                    ; (20/09/2022 - 27/06/2023) MSDOS 6.21 .. PCDOS 7.1
14730 000008A6 C3                                retn
14731                                  %endif
14732                                          ;;;
14733
14734                                  ; ---------------------------------------------------------------------------
14735
14736                                  mmi_not_ext:
14737 000008A7 F9                                stc
14738 000008A8 C3                                retn
14739
14740                                  ; =============== S U B   R O U T I N E =======================================
14741
14742                                  ; 15/10/2022
14743                                  ; ---------------------------------------------------------------
14744                                  ; read in the fat sector and get the media byte from it.
14745                                  ; input : es:di -> bds
14746                                  ; output:
14747                                  ;     carry set if an error occurs, ax contains error code.
14748                                  ;     otherwise, ah contains media byte on exit
14749                                  ; ---------------------------------------------------------------
14750
14751                                  readfat:
14752                                          ;mov    dh, 0
14753                                          ; 10/12/2022
14754 000008A9 30F6                              xor     dh, dh
14755 000008AB B90200                            mov     cx, 2           ; head 0
14756                                                                    ; cylinder 0, sector 2
14757 000008AE E80500                            call    read_sector
```

```
14758 000008B1 7202                             jb      short bad_fat_ret
14759 000008B3 8A27                             mov     ah, [bx]        ; media byte
14760                                   bad_fat_ret:
14761 000008B5 C3                               retn
14762
14763                                   ; =============== S U B   R O U T I N E =======================================
14764
14765                                   ; 15/10/2022
14766
14767                                   ; ---------------------------------------------------------------------------
14768                                   ; read a single sector into the temp buffer.
14769                                   ; perform three retries in case of error.
14770                                   ;   inputs: es:[di].bds_drivenum has physical drive to use
14771                                   ;           cx has sector and cylinder
14772                                   ;           dh has head
14773                                   ;           es:di has bds
14774                                   ;           ds has Bios_Data
14775                                   ;
14776                                   ;   outputs:        carry clear
14777                                   ;                Bios_Data:bx point to sector
14778                                   ;                   (note: some callers assume location of buffer)
14779                                   ;
14780                                   ;           carry set
14781                                   ;               ax has rom error code
14782                                   ;
14783                                   ; register bp is preserved.
14784                                   ; ---------------------------------------------------------------------------
14785
14786                                   ; 10/03/2019 - Retro DOS v4.0
14787                                   ; 22/12/2023 - Retro DOS v5.0
14788
14789                                               ; 19/10/2022
14790                                   read_sector:
14791 000008B6 55                               push    bp
14792 000008B7 BD0300                           mov     bp, 3           ; make 3 attempts
14793 000008BA 268A5504                         mov     dl, [es:di+4]   ; [es:di+BDS.drivenum]
14794 000008BE BB[5201]                         mov     bx, disksector ; get es:bx to point to      buffer
14795                                   rd_ret:
14796 000008C1 06                               push    es
14797 000008C2 1E                               push    ds
14798 000008C3 07                               pop     es
14799 000008C4 B80102                           mov     ax, 201h
14800 000008C7 CD13                             int     13h             ; DISK - READ SECTORS INTO MEMORY
14801                                                                   ; AL = number of sectors to read, CH = track, CL = sector
14802                                                                   ; DH = head, DL       = drive, ES:BX -> buffer to fill
14803                                                                   ; Return: CF set on error, AH =      status, AL = number of sectors read
14804 000008C9 07                               pop     es
14805 000008CA 734A                             jnb     short okret2
14806                                   rd_rty:
14807 000008CC E81205                           call    again           ; reset disk, decrement       bp, preserve ax
14808 000008CF 7442                             jz      short err_rd_ret
14809
14810                                                   ; 22/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
14811 000008D1 26F6453F01                       test    byte [es:di+3Fh], 1
14812                                           ;test   byte [es:di+23h], 1
14813                                           ;;test  byte ptr [es:di+23h], 1       ; [es:di+BDS.flags]
14814                                                                   ; fnon_removable
14815 000008D6 75E9                             jnz     short rd_ret
14816 000008D8 803E[A905]00                     cmp     byte [media_set_for_format], 0
14817 000008DD 7510                             jnz     short rd_skip1_dpt
14818 000008DF 50                               push    ax
14819 000008E0 1E                               push    ds              ; for retry, set the head settle time to 0Fh
14820 000008E1 C536[2D01]                       lds     si, [dpt]
14821                                           ;mov    al, [si+9]      ; [si+DISK_PARMS.DISK_HEAD_STTL]
14822                                           ;mov    byte [si+9], 15 ; [si+DISK_PARMS.DISK_HEAD_STTL]
14823                                                                   ; NORMSETTLE
14824                                                   ; 12/12/2022
14825 000008E5 B00F                             mov     al, 15
14826 000008E7 864409                           xchg    al, [si+9]
14827                                           ;
14828 000008EA 1F                               pop     ds
14829 000008EB A2[2A01]                         mov     [save_head_sttl], al
14830 000008EE 58                               pop     ax
14831                                   rd_skip1_dpt:
14832 000008EF 06                               push    es
14833 000008F0 1E                               push    ds
14834 000008F1 07                               pop     es
14835 000008F2 B80102                           mov     ax, 201h
14836 000008F5 CD13                             int     13h             ; DISK - READ SECTORS INTO MEMORY
14837                                                                   ; AL = number of sectors to read, CH = track, CL = sector
14838                                                                   ; DH = head, DL       = drive, ES:BX -> buffer to fill
14839                                                                   ; Return: CF set on error, AH =      status, AL = number of sectors read
14840 000008F7 07                               pop     es
14841 000008F8 9C                               pushf
14842 000008F9 803E[A905]00                     cmp     byte [media_set_for_format], 0
14843 000008FE 750E                             jnz     short rd_skip2_dpt
14844 00000900 50                               push    ax
14845 00000901 A0[2A01]                         mov     al, [save_head_sttl]
14846 00000904 1E                               push    ds
14847 00000905 C536[2D01]                       lds     si, [dpt]
14848 00000909 884409                           mov     [si+9], al      ; [si+DISK_PARMS.DISK_HEAD_STTL]
14849 0000090C 1F                               pop     ds
14850 0000090D 58                               pop     ax
14851                                   rd_skip2_dpt:
14852 0000090E 9D                               popf
14853 0000090F 7305                             jnb     short okret2
14854 00000911 EBB9                             jmp     short rd_rty
14855                                   ; ---------------------------------------------------------------------------
14856
14857                                   err_rd_ret:
14858 00000913 B2FF                             mov     dl, 0FFh        ; make sure we ask rom if media      has changed
14859                                                                   ; return error
14860 00000915 F9                               stc
14861
14862                                   ; update information pertaining to last drive accessed, time of access, last
14863                                   ; track accessed in that drive.
14864
14865                                   okret2:
14866 00000916 8816[7600]                       mov     [step_drv], dl ; set up for head settle logic in disk
14867 0000091A 8816[1E01]                       mov     [tim_drv], dl  ; save drive last accessed
14868
14869                                                   ; 22/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
14870 0000091E 26886D78                         mov     [es:di+78h], ch
14871                                           ;mov    [es:di+46h], ch       ; [es:di+BDS.track]
14872                                                                   ; save last track accessed on this drive
14873                                                                   ; preserve flags in case error occurred
14874 00000922 9C                               pushf
14875 00000923 E89B04                           call    set_tim
14876 00000926 9D                               popf                    ; restore flags
14877 00000927 5D                               pop     bp
14878 00000928 C3                               retn
14879
14880                                   ;---------------------------------------------------------------------------
14881                                   ;   disk open/close routines
```

```
14882                                   ;--------------------------------------------------------------------------
14883                                   
14884                                   dsk_open:                       ; 2C7h:80Ah = 70h:2D7Ah
14885 00000929 803E[7700]00                       cmp     byte [fhave96], 0
14886 0000092E 7407                               jz      short dsk_open_exit ; done if no changeline support
14887 00000930 E871FC                             call    SetDrive        ; get bds for drive
14888                                             ; 22/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
14889 00000933 26FF453C                           inc     word [es:di+3Ch] ; [es:di+BDS.opcnt] ; BDS offset 60
14890                                             ;inc    word [es:di+20h] ; [es:di+BDS.opcnt]
14891                                   dsk_open_exit:
14892                                             ; 10/12/2022
14893                                             ; cf = 0
14894                                             ;clc            ; CF is already ZERO here (18/09/2022, MSDOS 5.0 IO.SYS)
14895                                                             ; (19/07/2019 -       Erdogan Tan - MSDOS 6.0       IO.SYS - retrodos4.s)
14896 00000937 C3                                 retn
14897                                   ; --------------------------------------------------------------------------
14898                                   
14899                                   dsk_close:                      ; 2C7h:81Ah = 70h:2D8Ah
14900 00000938 803E[7700]00                       cmp     byte [fhave96], 0
14901 0000093D 740E                               jz      short exitjx   ; done if no changeline      support
14902 0000093F E862FC                             call    SetDrive        ; get bds for drive
14903                                             ; 22/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
14904 00000942 26837D3C00                         cmp     word [es:di+3Ch], 0 ; [es:di+BDS.opcnt] ; BDS off 60
14905                                             ;cmp    word [es:di+20h], 0 ; [es:di+BDS.opcnt]
14906 00000947 7404                               jz      short exitjx   ; watch out for wrap
14907                                             ; 22/12/2023
14908 00000949 26FF4D3C                           dec     word [es:di+3Ch]
14909                                             ;dec    word [es:di+20h]
14910                                   exitjx:
14911                                             ; 10/12/2022
14912                                             ; cf = 0
14913                                             ;clc            ; CF is already ZERO here (18/09/2022, MSDOS 5.0 IO.SYS)
14914                                                             ; (19/07/2019 -       Erdogan Tan - MSDOS 6.0       IO.SYS - retrodos4.s)
14915 0000094D C3                                 retn
14916                                   
14917                                   ;--------------------------------------------------------------------------
14918                                   ;       disk removable routine
14919                                   ;--------------------------------------------------------------------------
14920                                   
14921                                             ; al is unit #
14922                                   dsk_rem:                        ; 2C7h:831h = 70h:2DA1h
14923 0000094E E853FC                             call    SetDrive        ; get bds for this drive
14924                                             ; 22/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
14925                                             ;test   byte [es:di+BDS.flags], fnon_removable
14926 00000951 26F6453F01                         test    byte [es:di+3Fh], 1 ; [es:di+BDS.flags], fnon_removable
14927 00000956 74F5                               jz      short exitjx
14928                                             ;test   byte [es:di+23h], 1 ; [es:di+BDS.flags], fnon_removable
14929                                             ;;jnz   short x_bus_exit ; non_rem
14930                                             ;jnz    short non_rem  ; 15/10/2022
14931                                             ;; 10/12/2022
14932                                             ;; cf = 0
14933                                             ;;clc                  ; CF is already ZERO here
14934                                             ;                       ; 15/10/2022
14935                                             ;retn
14936                                   ; --------------------------------------------------------------------------
14937                                   
14938                                   non_rem:
14939                                   x_bus_exit:
14940 00000958 B403                               mov     ah, 3           ; 2C7h:83Dh = 0070h:2DADh
14941                                                                     ; return busy status
14942 0000095A F9                                 stc
14943                                   dsk_ret:
14944 0000095B C3                                 retn
14945                                   
14946                                   ;--------------------------------------------------------------------------
14947                                   ;       disk i/o routines
14948                                   ;--------------------------------------------------------------------------
14949                                   
14950                                   dsk_writv:                      ; 2C7h:841h = 70h:2DB1h
14951                                             ;mov    word [wrtverify], 103h
14952                                             ; 19/10/2022
14953 0000095C C706[2001]0301                     mov     word [rflag], 103h
14954                                             ;mov    word ptr ds:rflag, 103h       ; write and verify
14955 00000962 EB06                               jmp     short dsk_cl
14956                                   ; --------------------------------------------------------------------------
14957                                   
14958                                   dsk_writ:                       ; 2C7h:849h = 70h:2DB9h
14959                                             ;mov    word [wrtverify], 3
14960                                             ; 19/10/2022
14961 00000964 C706[2001]0300                     mov     word [rflag], 3
14962                                             ;mov    word ptr ds:rflag, 3 ; romwrite
14963                                   dsk_cl:
14964 0000096A E8A400                             call    diskio          ; romwrite
14965                                   ; --------------------------------------------------------------------------
14966                                   
14967                                   dsk_io:
14968 0000096D 73EC                               jnb     short dsk_ret
14969 0000096F E965F7                             jmp     bc_err_cnt
14970                                   ; --------------------------------------------------------------------------
14971                                   
14972                                   dsk_read:                       ; ; 2C7h:857h =       70h:2DC7h
14973 00000972 E89700                             call    diskrd
14974 00000975 EBF6                               jmp     short dsk_io
14975                                   
14976                                   ; =============== S U B   R O U T I N E =======================================
14977                                   
14978                                   ; 15/10/2022
14979                                   ; 10/03/2019 - Retro DOS v4.0
14980                                   ; 22/12/2023 - Retro DOS v5.0
14981                                   
14982                                   ;--------------------------------------------------------
14983                                   ; miscellaneous odd jump routines.
14984                                   ; moved out of mainline for speed.
14985                                   
14986                                   ; if we have a system where we have virtual drives, we need
14987                                   ; to prompt the user to place the correct disk in the drive.
14988                                   ;
14989                                   ;   assume es:di -> bds, ds:->Bios_Data
14990                                   ;--------------------------------------------------------
14991                                   
14992                                             ; 19/10/2022
14993                                   checksingle:
14994 00000977 50                                 push    ax
14995 00000978 53                                 push    bx
14996                                             ; 22/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
14997 00000979 268B5D3F                           mov     bx, [es:di+3Fh]       ; [es:di+BDS.flags]
14998                                             ;mov    bx, [es:di+23h]       ; [es:di+BDS.flags]
14999                                   
15000                                   ; if hard drive, cannot change disk.
15001                                   ; if current owner of physical drive, no need to change diskette.
15002                                   
15003 0000097D F6C321                             test    bl, 21h         ; fnon_removable|fi_own_physical
15004 00000980 7573                               jnz     short singleret
15005 00000982 F6C310                             test    bl, 10h         ; fi_am_mult
```

```
15006                                                   ; is there a drive sharing this      physical drive?
15007 00000985 746E                             jz      short singleret
15008
15009                                   ; look for the previous owner of this physical drive
15010                                   ; and reset its ownership flag.
15011
15012 00000987 268A4504                         mov     al, [es:di+4]  ; [es:di+BDS.drivenum]
15013                                                   ; get physical drive number
15014 0000098B 06                               push    es             ; preserve pointer to current bds
15015 0000098C 57                               push    di
15016 0000098D C43E[1901]                       les     di, [start_bds] ; get first bds
15017                                   scan_list:
15018 00000991 26384504                         cmp     [es:di+4], al
15019 00000995 7553                             jnz     short scan_skip       ; Not our drive. Try next bds.
15020 00000997 B320                             mov     bl, 20h ; ' '   ; fi_own_physical ; test ownership flag
15021                                           ; 22/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
15022 00000999 26845D3F                         test    [es:di+3Fh], bl       ; [es:di+BDS.flags]
15023                                           ;test   [es:di+23h], bl
15024 0000099D 744B                             jz      short scan_skip       ; he doesn't own it either. continue
15025 0000099F 26305D3F                         xor     [es:di+3Fh], bl
15026                                           ;xor    [es:di+23h], bl       ; reset ownership flag
15027 000009A3 5F                               pop     di             ; restore pointer to current bds
15028 000009A4 07                               pop     es
15029 000009A5 26085D3F                         or      [es:di+3Fh], bl
15030                                           ;or     [es:di+23h], bl       ; ; set ownership flag
15031
15032                                   ; we examine the fsetowner flag. if it is set, then we are using the code in
15033                                   ; checksingle to just set the owner of a drive. we must not issue the prompt
15034                                   ; in this case.
15035 000009A9 803E[7A00]01                     cmp     byte [fsetowner], 1
15036 000009AE 7517                             jnz     short not_fsetowner
15037                                           ;cmp    byte ptr es:[di+4], 0 ;       are we handling       drive number 0 ?
15038 000009B0 26807D0400                       cmp     byte [es:di+4], 0
15039 000009B5 753E                             jnz     short singleret
15040 000009B7 268A4505                         mov     al, [es:di+5]
15041                                           ;mov    al, es:[di+5]  ; [es:di+BDS.drivelet]
15042                                                   ; get the DOS drive letter
15043 000009BB 06                               push    es
15044 000009BC 8E06[1A00]                       mov     es, [zeroseg]
15045 000009C0 26A20405                         mov     [es:LSTDRV], al
15046                                           ;mov    es:504h, al    ; [es:LSTDRV]
15047                                                   ; set up sdsb
15048 000009C4 07                               pop     es             ; restore bds pointer
15049 000009C5 EB2E                             jmp     short singleret
15050                                   ; ---------------------------------------------------------------------------
15051
15052                                   ; to support "backward" compatibility with ibm's "single drive status byte"
15053                                   ; we now check to see if we are in a single drive system and the application
15054                                   ; has "cleverly" diddled the sdsb
15055
15056                                   not_fsetowner:
15057 000009C7 803E[7800]02                     cmp     byte [single], 2 ; if (single_drive_system)
15058 000009CC 7517                             jnz     short ignore_sdsb
15059 000009CE 50                               push    ax
15060 000009CF 268A4505                         mov     al, [es:di+5]  ; if (curr_drv == req_drv)
15061 000009D3 88C4                             mov     ah, al
15062 000009D5 06                               push    es
15063 000009D6 8E06[1A00]                       mov     es, [zeroseg]
15064 000009DA 2686060405                       xchg    al, [es:LSTDRV]
15065                                           ;xchg   al, es:504h    ; [es:LSTDRV]
15066                                                   ; then swap(curr_drv,req_drv)
15067 000009DF 07                               pop     es
15068 000009E0 38C4                             cmp     ah, al         ; else
15069 000009E2 58                               pop     ax             ; swap(curr_drv,req_drv)
15070 000009E3 7410                             jz      short singleret       ; issue swap_dsk_msg
15071                                   ignore_sdsb:
15072 000009E5 E8B310                           call    swpdsk
15073 000009E8 EB0B                             jmp     short singleret
15074                                   ; ---------------------------------------------------------------------------
15075
15076                                   scan_skip:
15077 000009EA 26C43D                           les     di, [es:di]
15078                                           ;les    di, es:[di]    ; [es:di+BDS.link]
15079                                                   ; go to next bds
15080 000009ED 83FFFF                           cmp     di, 0FFFFh ; -1       ; end of list?
15081 000009F0 759F                             jnz     short scan_list       ; continue until hit end of list
15082 000009F2 F9                               stc
15083 000009F3 5F                               pop     di             ; restore current bds
15084 000009F4 07                               pop     es
15085                                   singleret:
15086 000009F5 5B                               pop     bx
15087 000009F6 58                               pop     ax
15088 000009F7 C3                               retn
15089
15090                                   ; 22/12/2023
15091                                   %if 0
15092                                   ; ---------------------------------------------------------------------------
15093
15094                                   baddrive:
15095                                           mov     al, 8          ; sector not found
15096                                           jmp     short baddrive_ret
15097                                   %endif
15098
15099                                   ; ---------------------------------------------------------------------------
15100
15101                                   unformatteddrive:
15102 000009F8 B007                             mov     al, 7          ; unknown media
15103                                   ;baddrive_ret:
15104 000009FA F9                               stc
15105                                   ; ---------------------------------------------------------------------------
15106
15107                                   ioret:
15108 000009FB C3                               retn
15109
15110                                   ; ---------------------------------------------------------------------------
15111
15112                                           ; 22/12/2023 - Retro DOS v5.0
15113                                           ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0A1Bh
15114
15115 000009FC 10                       LBA_Packet: db 16              ; ...
15116                                                                  ; DAP buffer
15117 000009FD 00                                   db 0
15118 000009FE 0000                     dap_block_cnt:     dw 0                ; ...
15119 00000A00 00000000                 dap_trans_buf:     dd 0                ; ...
15120 00000A04 00000000                 dap_lba_value:     dd 0                ; ...
15121 00000A08 00000000                         dd 0
15122
15123                                   ; ---------------------------------------------------------------------------
15124
15125                                   ; 15/10/2022
15126
15127                                   ; ---------------------------------------------------------------------------
15128                                   ;   disk i/o handler
15129                                   ;
```

```
15130                                   ;   al = drive number (0-6)
15131                                   ;   ah = media descriptor
15132                                   ;   cx = sector count
15133                                   ;   dx = first sector (low)
15134                                   ;   [start_sec_h] = first sector (high)  32 bit calculation.
15135                                   ;   ds = cs
15136                                   ;   es:di = transfer address
15137                                   ;   [rflag]=operation (2=read, 3=write)
15138                                   ;   [verify]=1 for verify after write
15139                                   ;
15140                                   ;   if successful carry flag = 0
15141                                   ;     else cf=1 and al contains error code
15142                                   ; ---------------------------------------------------------------------------
15143
15144                                           ; 12/12/2023
15145                                           ; ds = biosdata segment (cs = bioscode segment)
15146                                   diskrd:
15147                                           ;mov    ds:rflag, 2    ; romread
15148                                           ; 19/10/2022
15149 00000A0C C606[2001]02                     mov     byte [rflag], 2 ; romread
15150
15151                                   ; =============== S U B   R O U TI N E =======================================
15152
15153                                           ; 22/12/2023 - Retro DOS v5.0
15154                                           ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:0A30h
15155                                   ; 22/12/2023
15156                                   %if 0
15157                                           ; 19/10/2022
15158                                   diskio:
15159                                           mov     bx, di          ; es:bx= transfer address
15160                                           mov     [xfer_seg], es ; save transfer       segment
15161                                           call    SetDrive
15162                                           mov     al, [es:di+10h]         ; [es:di+BDS.media]
15163                                           mov     [medbyt], al
15164                                           ;jcxz   short ioret
15165                                           jcxz    ioret
15166
15167                                   ; see if the media is formatted or not by checking the flags field in
15168                                   ; in the bds. if it is unformatted we cannot allow i/o, so we should
15169                                   ; go to the error exit at label unformatteddrive.
15170
15171                                           test    byte [es:di+24h], 2
15172                                           ;test   byte ptr es:[di+24h], 2       ; [es:di+BDS.flags+1]
15173                                                                   ; unformatted_media
15174                                           jnz     short unformatteddrive
15175                                           mov     [seccnt], cx    ; save sector count
15176                                           mov     [spsav], sp     ; save sp
15177
15178                                   ; ensure that we are trying to access valid sectors on the drive
15179
15180                                           mov     ax, dx
15181                                           xor     si, si ; 0
15182                                           add     dx, cx
15183                                           ;adc    si, 0
15184                                           ; 02/09/2023 (PCDOS 7.1)
15185                                           rcl     si, 1
15186                                           cmp     word [es:di+0Eh], 0 ; [es:di+BDS.totalsecs16]
15187                                                                   ; 32 bit sector ?
15188                                           jz      short sanity32
15189                                           ;cmp    si, 0
15190                                           ; 02/09/2023
15191                                           or      si, si
15192                                           jnz     short baddrive
15193                                           cmp     dx, [es:di+0Eh]         ; [es:di+BDS.totalsecs16]
15194                                           ja      short baddrive
15195                                           jmp     short sanityok
15196                                   ; ---------------------------------------------------------------------------
15197
15198                                   sanity32:
15199                                           add     si, [start_sec_h]
15200                                           cmp     si, [es:di+1Dh]         ; [es:di+BDS.totalsecs32+2]
15201                                           jb      short sanityok
15202                                           ja      short baddrive
15203                                           cmp     dx, [es:di+1Bh]         ; [es:di+BDS.totalsecs32]
15204                                           ja      short baddrive
15205                                   sanityok:
15206                                           mov     dx, [start_sec_h]
15207                                           add     ax, [es:di+17h]         ; [es:di+BDS.hiddensecs]
15208                                           adc     dx, [es:di+19h]         ; [es:di+BDS.hiddensecs+2]
15209
15210                                   ; now dx;ax have the physical first sector.
15211                                   ; since the following procedures is going to destroy ax, let's
15212                                   ; save it temporarily to saved_word.
15213
15214                                           mov     [saved_word], ax ; save the sector number (low)
15215
15216                                   ; set up pointer to disk base table in [dpt]. we cannot assume that iosetup
15217                                   ; will do it because we will skip the set up stuff with hard disks.
15218
15219                                           push    es
15220                                           ;mov    es, [zeroseg]
15221                                           ; 02/09/2023
15222                                           xor     si, si ; 0
15223                                           mov     es, si
15224                                           les     si, [es:DSKADR]
15225                                           ;les    si, es:78h      ; [es:DSKADR]
15226                                                                   ; current disk parm table
15227                                           mov     [dpt], si
15228                                           mov     [dpt+2], es
15229                                           pop     es
15230                                           test    byte [es:di+23h], 1 ; [es:di+BDS.flags]
15231                                                                   ; fnon_removable
15232                                           jnz     short skip_setup
15233                                           call    checksingle
15234
15235                                   ; check to see if we have previously noted a change line. the routine
15236                                   ; returns if everything is ok. otherwise, it pops off the stack and returns
15237                                   ; the proper error code.
15238
15239                                           cmp     byte [fhave96], 0 ; do we have changeline support?
15240                                           jz      short diskio_nochangeline ; brif not
15241                                           call    checklatchio    ; will do a sneaky pop stack return
15242                                                                   ; if a disk error occurs
15243                                   diskio_nochangeline:
15244                                           call    iosetup         ; set up tables       and variables for i/o
15245
15246                                   ; now the settle values are correct for the following code
15247
15248                                   skip_setup:
15249
15250                                   ; 32 bit sector calculation.
15251                                   ; dx:[saved_word] = starting sector number.
15252
15253                                           mov     ax, dx
```

```
                                        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
                                        ;les    si, dword ptr es:78h
00000A75 26C4367800                     les     si, [es:78h]    ; INT 1Eh vector address
                                                                ; [es:DSKADR] - current disk parm table
00000A7A 8936[2D01]                     mov     [dpt], si
00000A7E 8C06[2F01]                     mov     [dpt+2], es
00000A82 07                             pop     es
00000A83 26F6453F01                     test    byte [es:di+3Fh], 1 ; [es:di+BDS.flags], fnon_removable
00000A88 7510                           jnz     short chk_13h_ext_flag
00000A8A E8EAFE                         call    checksingle

                                ; check to see if we have previously noted a change line. the routine
                                ; returns if everything is ok. otherwise, it pops off the stack and returns
                                ; the proper error code.

00000A8D 803E[7700]00                   cmp     byte [fhave96], 0 ; do we have changeline support?
00000A92 7403                           jz      short diskio_nochangeline ; brif not
00000A94 E8D210                         call    checklatchio    ; will do a sneaky pop stack return
                                                                ; if a disk error occurs
                                diskio_nochangeline:
00000A97 E8E000                         call    iosetup         ; set up tables and variables for i/o

                                chk_13h_ext_flag:
00000A9A 26F6454004                     test    byte [es:di+40h], 4 ; [es:di+BDS.flags+1], fLBArw
                                                                ; LBA read/write flag
00000A9F 7539                           jnz     short set_lbarw_1
                                        ;jmp    skip_setup
                                        ; 22/12/2023
                                ; ---------------------------------------------------------------------------

                                ; now the settle values are correct for the following code

                                skip_setup:

                                ; 32 bit sector calculation.
                                ; dx:[saved_word] = starting sector number.

                                        ;push   bp ; ! (not necessary) ; 22/12/2023
00000AA1 92                             xchg    ax, dx ; mov ax,dx
00000AA2 31D2                           xor     dx, dx
00000AA4 268B4D13                       mov     cx, [es:di+13h]         ; [es:di+BDS.secpertrack]
                                                                ; divide by sec per track
00000AA8 F7F1                           div     cx
00000AAA 95                             xchg    ax, bp ; mov bp,ax
00000AAB A1[9E04]                       mov     ax, [saved_word]
00000AAE F7F1                           div     cx              ; [es:di+BDS.secpertrack]
                                                                ; now, bp:ax = track #, dx = sector
                                                                ; sector number is 1 based.
00000AB0 42                             inc     dx
00000AB1 8816[3101]                     mov     [cursec], dl    ; save current sector
00000AB5 268B4D15                       mov     cx, [es:di+15h] ; [es:di+BDS.heads]
                                                                ; get number of heads
                                        ; 22/12/2023
                                        ;push   ax ; *
00000AB9 31D2                           xor     dx, dx
00000ABB 95                             xchg    ax, bp ; bp = *         ; divide tracks by heads per cylinder
00000ABC F7F1                           div     cx
00000ABE 95                             xchg    ax, bp ; ax = *, bp = **
                                        ;pop    ax ; *
00000ABF F7F1                           div     cx              ; now, bp:ax = cylinder #, dx = head
00000AC1 09ED                           or      bp, bp ; ** = 0 ?
                                        ;pop    bp ; ! ; 22/12/2023
                                        ;jnz    short baddrive_brdg
                                        ; 22/12/2023
00000AC3 7586                           jnz     short baddrive

                                        ;cmp    ax, 1024        ; 2^10 currently maximum for track #.
                                        ;jnb    short baddrive_brdg
                                        ; 22/12/2023
00000AC5 80FC04                         cmp     ah, 4   ; if ax >= 4*256 (1024)
00000AC8 7381                           jnb     short baddrive

00000ACA 8816[3201]                     mov     [curhd], dl     ; save current head
00000ACE A3[3301]                       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.

00000AD1 A1[2201]                       mov     ax, [seccnt]
00000AD4 E81F01                         call    block
                                        ;call   done
                                        ;retn
                                        ; 22/12/2023
00000AD7 E9E500                         jmp     done

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

                                set_lbarw_1:
00000ADA A1[9E04]                       mov     ax, [saved_word] ; check for mini disk
                                                                 ; (logical dos drive/partition)
00000ADD 26837D7901                     cmp     word [es:di+79h], 1 ; [di+BDS.bdsm_ismini]
                                                                  ; logical dos partition
00000AE2 750F                           jnz     short set_lbarw_2 ; not a logical dos partition/drive
00000AE4 26837D7B00                     cmp     word [es:di+7Bh], 0 ; [di+BDS.bdsm_hidden_trks] (> 0)
00000AE9 7408                           jz      short set_lbarw_2
00000AEB 26034517                       add     ax, [es:di+17h]         ; [es:di+BDS.hiddensecs]
00000AEF 26135519                       adc     dx, [es:di+19h]         ; [es:di+BDS.hiddensecs+2]

                                set_lbarw_2:
00000AF3 2EA3[040A]                     mov     [cs:dap_lba_value], ax
00000AF7 2E8916[060A]                   mov     [cs:dap_lba_value+2], dx
00000AFC 2E891E[000A]                   mov     [cs:dap_trans_buf], bx
00000B01 A1[A804]                       mov     ax, [xfer_seg]
00000B04 2EA3[020A]                     mov     [cs:dap_trans_buf+2], ax
00000B08 A1[2201]                       mov     ax, [seccnt]
00000B0B 2EA3[FE09]                     mov     [cs:dap_block_cnt], ax
00000B0F BD0500                         mov     bp, 5
00000B12 892E[A304]                     mov     [vretry_cnt], bp ; verify op. retry cnt for write-verify
00000B16 892E[A504]                     mov     [soft_ecc_cnt], bp ; soft ecc error retry count

                                set_lbarw_3:
00000B1A 268A5504                       mov     dl, [es:di+4]  ; [es:di+BDS.drivenum]
00000B1E 8A26[2001]                     mov     ah, [rflag]    ; get read/write indicator
00000B22 80C440                         add     ah, 40h
00000B25 30C0                           xor     al, al
00000B27 1E                             push    ds
00000B28 0E                             push    cs
00000B29 1F                             pop     ds
00000B2A BE[FC09]                       mov     si, LBA_Packet
00000B2D CD13                           int     13h             ; LBA read/write
00000B2F 1F                             pop     ds
00000B30 731A                           jnc     short set_lbarw_7
00000B32 E8AC02                         call    again
```

```
                                set_lbarw_9:
00000B35 7503                           jnz     short set_lbarw_4
00000B37 E92B02                         jmp     harderr
                                ; ---------------------------------------------------------------------------

                                set_lbarw_4:
                                ;set_lbarw_9:      ; 22/12/2023
00000B3A 80FCCC                         cmp     ah, 0CCh        ; Write fault (hard disk)
00000B3D 7505                           jnz     short set_lbarw_5
00000B3F BD0100                         mov     bp, 1
                                        ;jmp    short set_lbarw_6
                                        ; 17/04/2024
00000B42 EBD6                           jmp     short set_lbarw_3
                                ; ---------------------------------------------------------------------------

                                set_lbarw_5:
                                set_lbarw_10:      ; 22/12/2023
00000B44 C706[A504]0500                 mov     word [soft_ecc_cnt], 5 ; soft ecc error retry count
                                set_lbarw_6:
                                set_lbarw_11:
00000B4A EBCE                           jmp     short set_lbarw_3
                                ; ---------------------------------------------------------------------------

                                set_lbarw_7:
00000B4C 813E[2001]0301                 cmp     word [rflag], 103h
00000B52 7523                           jnz     short set_lbarw_12
00000B54 B444                           mov     ah, 44h
00000B56 1E                             push    ds
00000B57 0E                             push    cs
00000B58 1F                             pop     ds
00000B59 CD13                           int     13h             ; DISK - IBM/MS Extension - VERIFY SECTORS
                                                                ;  (DL - drive, [SI - disk address packet)
00000B5B 1F                             pop     ds
00000B5C 7319                           jnc     short set_lbarw_12
00000B5E 80FC11                         cmp     ah, 11h         ; ECC corrected data error (soft error - retried OK )
00000B61 7506                           jnz     short set_lbarw_8
00000B63 FF0E[A504]                     dec     word [soft_ecc_cnt]
                                ;set_lbarw_8:
00000B67 740E                           jz      short set_lbarw_12
                                set_lbarw_8:
00000B69 E8CF07                         call    ResetDisk
00000B6C 80FC11                         cmp     ah, 11h
00000B6F 74D9                           jz      short set_lbarw_11
00000B71 FF0E[A304]                     dec     word [vretry_cnt]
                                        ;jnz    short set_lbarw_9
                                        ;jmp    harderr
                                        ; 22/12/2023
00000B75 EBBE                           jmp     short set_lbarw_9

                                ; ---------------------------------------------------------------------------
                                ;               ; 22/12/2023
                                ;set_lbarw_9:
                                ;               cmp     ah, 0CCh
                                ;               jnz     short set_lbarw_10
                                ;               mov     bp, 1
                                ;               jmp     short set_lbarw_11
                                ; ---------------------------------------------------------------------------
                                ;               ; 22/12/2023
                                ;set_lbarw_10:
                                ;               mov     word [soft_ecc_cnt], 5 ; soft ecc error retry count
                                ;set_lbarw_11:
                                ;               jmp     short set_lbarw_3
                                ; ---------------------------------------------------------------------------

                                set_lbarw_12:
00000B77 31C0                           xor     ax, ax
                                skip_dpt_setting:  ; 23/12/2023
00000B79 C3                             retn
                                        ;;;     ; 22/12/2023
                                %endif

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

                                        ; 22/12/2023
                                ;baddrive_brdg:
                                        ;jmp    baddrive

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

                                ;---------------------------------------------------------------
                                ; set the drive-last-accessed flag for diskette only.
                                ; we know that the hard disk will not be removed.
                                ; es:di -> current bds.
                                ; ds -> Bios_Data
                                ; ax,cx,si are destroyed.
                                ;---------------------------------------------------------------

                                        ; 23/12/2023 - Retro DOS v5.0

                                        ; 19/10/2022
                                iosetup:
00000B7A 268A4504                       mov     al, [es:di+4]  ; [es:di+BDS.drivenum]
00000B7E A2[1E01]                       mov     [tim_drv], al  ; save drive letter

                                ; determine proper head settle values

00000B81 803E[A905]00                   cmp     byte [media_set_for_format], 0
00000B86 75F1                           jnz     short skip_dpt_setting
00000B88 A0[2C01]                       mov     al, [eot]       ; fetch up eot before changing ds
00000B8B 1E                             push    ds
00000B8C C536[2D01]                     lds     si, [dpt]       ; get pointer to disk base table
00000B90 884404                         mov     [si+4],al

                                        ;; 23/12/2023
                                        ;mov    ah, al
                                        ;mov    al, [si+10]     ; [si+DISK_PARMS.DISK_MOTOR_STRT]
                                        ;;mov   ah, [si+4]      ; [si+DISK_PARMS.DISK_EOT]
                                        ;pop    ds
                                        ;mov    [motorstartup], al
                                        ;mov    [save_eot], ah
                                        ; 06/04/2024
00000B93 8A640A                         mov     ah,[si+10]
00000B96 1F                             pop     ds
00000B97 8826[2601]                     mov     [motorstartup], ah
00000B9B A2[2B01]                       mov     [save_eot], al

                                ; for 3.5" drives, both external as well as on the k09, we need to set the
                                ; motor start time to 4. this checking for every i/o is going to affect
                                ; performance across the board, but is necessary!!

00000B9E 1E                             push    ds
00000B9F C536[2D01]                     lds     si, [dpt]       ; get pointer to disk base table
                                        ; 23/12/2023  - Retro DOS v5.0
00000BA3 26807D3E02                     cmp     byte [es:di+3Eh], 2 ; (PCDOS 7.1 IBMBIO.COM)
```

```
                                        ;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 T I 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 T I 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
```

```
15750
15751 00000C28 29C8                           sub     ax, cx          ; reduce sectors-remaining by last i/o
15752 00000C2A D0E1                           shl     cl, 1
15753 00000C2C 00CF                           add     bh, cl          ; adjust transfer address
15754 00000C2E EBC6                           jmp     short block
15755                                 dskerr_brdg:
15756 00000C30 E9F400                         jmp     dskerr
15757
15758                                 ; =============== S U B   R O U TI N E =======================================
15759
15760                                 ; 15/10/2022
15761
15762                                 ;--------------------------------------------------------------
15763                                 ;perform disk i/o with retries
15764                                 ; al = number of sectors (1-8, all on one track)
15765                                 ; es:di point to drive parameters
15766                                 ; xfer_seg:bx = transfer address
15767                                 ;               (must not cross a 64k physical boundary)
15768                                 ; [rflag] = 2 if read, 3 if write
15769                                 ; [verify] = 0 for normal, 1 for verify after write
15770                                 ;--------------------------------------------------------------
15771
15772                                           ; 18/04/2024
15773                                           ; 23/12/2023 - Retro DOS v5.0
15774                                           ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0C74h)
15775
15776                                           ; 19/10/2022
15777                                 Disk:
15778
15779                                 ; Check for hard disk format and
15780                                 ; if TRUE then set max error count to 2
15781
15782 00000C33 BD0500                         mov     bp, 5           ; MAXERR
15783                                                                   ; set up retry count
15784                                           ; 18/04/2024
15785                                           ;; 23/12/2023
15786                                           ;mov    cl, [es:di+3Fh]
15787                                           ;and    cx, 1
15788 00000C36 26F6453F01                       test    byte [es:di+3Fh], 1
15789                                           ;test   byte [es:di+23h], 1
15790                                                                   ; [es:di+BDS.flags], fnon_removable
15791 00000C3B 7408                           jz      short GetRdWrInd
15792 00000C3D 80FC04                         cmp     ah, 4           ; romverify ; Is this a       track verify?
15793 00000C40 7403                           jz      short GetRdWrInd
15794 00000C42 BD0200                         mov     bp, 2           ; This is not verify so only 1 retry
15795                                 GetRdWrInd:
15796 00000C45 892E[A304]                     mov     [vretry_cnt], bp ; verify op. retry cnt for write-verify
15797 00000C49 892E[A504]                     mov     [soft_ecc_cnt], bp ; soft ecc error retry count.
15798 00000C4D 8A26[2001]                     mov     ah, [rflag]     ; get read/write indicator
15799                                 ;retry:
15800                                 ; 09/12/2022
15801                                 _retry:
15802 00000C51 50                             push    ax
15803 00000C52 8B16[3301]                     mov     dx, [curtrk]
15804                                           ;; 23/12/2023
15805                                           ;jcxz   disk_not_mini
15806                                           ; 18/04/2024
15807 00000C56 26F6453F01                     test    byte [es:di+3Fh], 1
15808                                           ;test   byte [es:di+23h], 1
15809 00000C5B 740B                           jz      short disk_not_mini
15810
15811                                           ; 23/12/2023
15812 00000C5D 26837D7901                     cmp     word [es:di+79h], 1
15813                                           ;cmp    word [es:di+47h], 1 ; [es:di+BDS.bdsm_ismini]
15814                                                                   ; is this a mini disk? ((logical dos partition))
15815 00000C62 7504                           jnz     short disk_not_mini ; no. continue to next.
15816                                           ; 23/12/2023
15817 00000C64 2603557B                       add     dx, [es:di+7Bh]
15818                                           ;add    dx, [es:di+49h]       ; [es:di+BDS.bdsm_hidden_trks]
15819                                                                   ; add hidden trks.
15820                                 disk_not_mini:
15821 00000C68 D0CE                           ror     dh, 1
15822 00000C6A D0CE                           ror     dh, 1
15823 00000C6C 0A36[3101]                     or      dh, [cursec]
15824 00000C70 89D1                           mov     cx, dx
15825 00000C72 86CD                           xchg    ch, cl          ;  cl = sector, ch = cylinder
15826 00000C74 8A36[3201]                     mov     dh, [curhd]     ; load current head number and
15827 00000C78 268A5504                       mov     dl, [es:di+4]   ; physical drive number
15828                                                                   ; [es:di+BDS.drivenum]
15829                                           ; 23/12/2023
15830 00000C7C 26807D3E05                     cmp     byte [es:di+3Eh], 5
15831                                           ;cmp    byte [es:di+22h], 5 ; [es:di+BDS.formfactor], ffHardFile
15832 00000C81 7411                           jz      short do_fast   ; hard files use fast speed
15833
15834                                 ; if we have [step_drv] set to -1, we use the slow settle time.
15835                                 ; this helps when we have just done a reset disk operation and the head has
15836                                 ; been moved to another cylinder - the problem crops up with 3.5" drives.
15837
15838 00000C83 803E[7600]FF                     cmp     byte [step_drv], 0FFh ; -1
15839                                           ;jz     short do_writej
15840                                           ; 23/12/2023
15841 00000C88 746A                           jz      short do_write
15842 00000C8A 80FC02                         cmp     ah, 2           ; romread
15843 00000C8D 7405                           jz      short do_fast
15844 00000C8F 80FC04                         cmp     ah, 4           ; romverify
15845                                           ;jz     short do_fast
15846                                           ; 23/12/2023
15847 00000C92 7560                           jnz     short do_write
15848                                 ;do_writej:
15849
15850                                 ; reads always fast, unless we have just done a disk reset operation
15851
15852                                           ;jmp    short do_write ; reads always fast
15853                                 ; ---------------------------------------------------------------------------
15854
15855                                 do_fast:
15856 00000C94 E80501                         call    fastspeed       ; change settle       mode
15857                                 testerr:
15858 00000C97 7297                           jb      short dskerr_brdg
15859
15860                                           ; 23/12/2023 Retro DOS v5.0
15861                                           ; (PCDOS 7.1 IBMBIO.COM)
15862 00000C99 83FD05                         cmp     bp, 5           ; is there retry ?
15863 00000C9C 7505                           jnz     short testerror       ; yes
15864 00000C9E 80FCBB                         cmp     ah, 0BBh        ; Undefined error (hard disk)
15865 00000CA1 748D                           jz      short dskerr_brdg
15866                                 testerror:
15867
15868                                 ; set drive and track of last access
15869
15870 00000CA3 8816[7600]                     mov     [step_drv], dl
15871                                           ; 23/12/2023
15872 00000CA7 26886D78                       mov     [es:di+78h], ch
15873                                           ;mov    [es:di+46h], ch       ; [es:di+BDS.track]
```

```
15874                                   no_set:
15875                                           ;cmp   word [wrtverify], 103h
15876 00000CAB 813E[2001]0301                   cmp    word [rflag], 103h ; check for write and verify
15877 00000CB1 7452                             jz     short doverify
15878                                   noverify:
15879 00000CB3 58                               pop    ax
15880
15881                                   ; check the flags word in the bds to see if the drive is non removable
15882                                   ; if not we needn't do anything special
15883                                   ; if it is a hard disk then check to see if multi-track operation
15884                                   ; is specified. if specified we don't have to calculate for the next
15885                                   ; track since we are already done. so we can go to the exit of this routine.
15886
15887                                           ; 23/12/2023
15888 00000CB4 26F6453F01                       test   byte [es:di+3Fh], 1
15889                                           ;test  byte [es:di+23h], 1 ; [es:di+BDS.flags]
15890                                                                      ; fnon_removable
15891 00000CB9 7407                             jz     short its_removable
15892 00000CBB F606[A004]80                     test   byte [multrk_flag], 80h ; multrk_on
15893 00000CC0 7530                             jnz    short disk_ret
15894                                   its_removable:
15895 00000CC2 80E13F                           and    cl, 3Fh        ; eliminate cylinder bits from sector
15896 00000CC5 30E4                             xor    ah, ah
15897 00000CC7 2906[2201]                       sub    [seccnt], ax   ; reduce count of sectors to go      next sector
15898 00000CCB 00C1                             add    cl, al
15899 00000CCD 880E[3101]                       mov    [cursec], cl
15900 00000CD1 263A4D13                         cmp    cl, [es:di+13h]       ; [es:di+BDS.secpertrack]
15901                                                                  ; see if sector/track limit reached
15902 00000CD5 761B                             jbe    short disk_ret
15903 00000CD7 C606[3101]01                     mov    byte [cursec], 1 ; start with first sector of next track
15904 00000CDC 8A36[3201]                       mov    dh, [curhd]
15905 00000CE0 FEC6                             inc    dh
15906 00000CE2 263A7515                         cmp    dh, [es:di+15h]       ; [es:di+BDS.heads]
15907 00000CE6 7206                             jb     short noxor
15908 00000CE8 30F6                             xor    dh, dh
15909 00000CEA FF06[3301]                       inc    word [curtrk]
15910                                   noxor:
15911 00000CEE 8836[3201]                       mov    [curhd], dh
15912                                   disk_ret:
15913 00000CF2 F8                               clc
15914 00000CF3 C3                               retn
15915                                   ; ---------------------------------------------------------------------------
15916
15917                                   ; 15/10/2022
15918
15919                                   ; 24/12/2023 - Retro DOS v5.0
15920
15921                                   ;--------------------------------------------------------------
15922                                   ; the request is for write. determine if we are talking about
15923                                   ; the same track and drive. if so, use the fast speed.
15924                                   ;--------------------------------------------------------------
15925
15926                                   do_write:
15927 00000CF4 3A16[7600]                       cmp    dl, [step_drv]
15928 00000CF8 7506                             jnz    short do_norm  ; we have changed drives
15929                                           ; 24/12/2023
15930 00000CFA 263A6D78                         cmp    ch, [es:di+78h]
15931                                           ;cmp   ch, [es:di+46h]       ; [es:di+BDS.track]
15932 00000CFE 7494                             jz     short do_fast  ; we are still on the same track
15933                                   do_norm:
15934 00000D00 E87500                           call   normspeed
15935 00000D03 EB92                             jmp    short testerr
15936                                   ; ---------------------------------------------------------------------------
15937
15938                                   ;--------------------------------------------------------------
15939                                   ; we have a verify request also. get state info and go verify
15940                                   ;--------------------------------------------------------------
15941
15942                                   doverify:
15943 00000D05 58                               pop    ax
15944 00000D06 50                               push   ax
15945 00000D07 B404                             mov    ah, 4
15946 00000D09 E89000                           call   fastspeed
15947 00000D0C 73A5                             jnb    short noverify
15948
15949                                   ; check the error returned in ah to see if it is a soft ecc error.
15950                                   ; if it is not we needn't do anything special. if it is a soft
15951                                   ; ecc error then decrement the soft_ecc_cnt error retry count. if
15952                                   ; this retry count becomes 0 then we just ignore the error and go to
15953                                   ; no_verify but if we can still try then we call the routine to reset
15954                                   ; the disk and go to dskerr1 to retry the operation.
15955
15956 00000D0E 80FC11                           cmp    ah, 11h        ; soft ecc error ?
15957 00000D11 750B                             jnz    short not_softecc_err
15958 00000D13 FF0E[A504]                       dec    word [soft_ecc_cnt]
15959 00000D17 749A                             jz     short noverify ; no more retry
15960 00000D19 E81F06                           call   ResetDisk      ; reset disk
15961 00000D1C EB3E                             jmp    short dskerr1  ; retry
15962                                   ; ---------------------------------------------------------------------------
15963
15964                                   not_softecc_err:                ; other error.
15965 00000D1E E81A06                           call   ResetDisk
15966 00000D21 FF0E[A304]                       dec    word [vretry_cnt]
15967 00000D25 EB1C                             jmp    short dskerr0
15968                                   ; ---------------------------------------------------------------------------
15969
15970                                   ;--------------------------------------------------------------
15971                                   ; need to special case the change-line error ah=06h.
15972                                   ; if we get this, we need to return it.
15973                                   ;--------------------------------------------------------------
15974
15975                                   dskerr:
15976 00000D27 803E[7700]00                     cmp    byte [fhave96], 0     ; do we have changeline       support?
15977 00000D2C 7403                             jz     short dskerr_nochangeline ; brif not
15978 00000D2E E8BE0E                           call   checkio
15979                                   dskerr_nochangeline:
15980 00000D31 803E[A704]01                     cmp    byte [multitrk_format_flag], 1 ; multi trk format request?
15981 00000D36 7508                             jnz    short dochkagain ; no more retry.
15982 00000D38 BD0100                           mov    bp, 1
15983 00000D3B C606[A704]00                     mov    byte [multitrk_format_flag], 0 ; clear the flag.
15984                                   dochkagain:
15985 00000D40 E89E00                           call   again
15986                                   dskerr0:
15987 00000D43 7420                             jz     short harderr
15988                                           ; 24/12/2023
15989 00000D45 26F6453F01                       test   byte [es:di+3Fh], 1
15990                                           ;test  byte [es:di+23h], 1 ; [es:di+BDS.flags]
15991                                                                      ; fnon_removable
15992 00000D4A 7505                             jnz    short skip_timeout_chk
15993 00000D4C 80FC80                           cmp    ah, 80h        ; timeout?
15994 00000D4F 7414                             jz     short harderr
15995                                   skip_timeout_chk:
15996 00000D51 80FCCC                           cmp    ah, 0CCh       ; write fault error?
15997 00000D54 740A                             jz     short write_fault_err ;       then, don't retry.
```

```
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
```

```
16122
16123                                   set_tim:
16124 00000DC1 50                                 push    ax
16125 00000DC2 E86CF7                             call    GetTickCnt      ; Does INT 1A ah=0 & updates daycnt
16126
16127                                   ; we have the new time. if we see that the time has passed,
16128                                   ; then we reset the threshold counter...
16129
16130                                             ; 24/12/2023 - Retro DOS v5.0
16131 00000DC5 263B5579                           cmp     dx, [es:di+79h]       ; PCDOS 7.1 IBMBIO.COM
16132                                             ;cmp    dx, [es:di+47h]       ; [es:di+BDS.tim_lo]
16133 00000DC9 7506                               jne     short setaccess
16134                                             ; 24/12/2023
16135 00000DCB 263B4D7B                           cmp     cx, [es:di+7Bh]       ; PCDOS 7.1 IBMBIO.COM
16136                                             ;cmp    cx, [es:di+49h]       ; [es:di+BDS.tim_hi]
16137                                             ;jz     short done_set
16138                                             ; 12/12/2022
16139 00000DCF 740E                               je      short done_set2
16140                                   setaccess:
16141 00000DD1 C606[1D01]00                       mov     byte [accesscount], 0
16142
16143                                             ; 24/12/2023 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
16144 00000DD6 26895579                           mov     [es:di+79h], dx
16145 00000DDA 26894D7B                           mov     [es:di+7Bh], cx
16146                                             ;mov    [es:di+47h], dx       ; [es:di+BDS.tim_lo]
16147                                             ;mov    [es:di+49h], cx       ; [es:di+BDS.tim_hi]
16148                                   done_set:
16149 00000DDE F8                                 clc
16150                                   done_set2:        ; 12/12/2022
16151 00000DDF 58                                 pop     ax
16152 00000DE0 C3                                 retn
16153
16154                                   ; =============== S U B   R O U T I N E =======================================
16155
16156                                   ; this routine is called if an error occurs while formatting or verifying.
16157                                   ; it resets the drive,and decrements the retry count.
16158                                   ; on entry - ds:di - points to bds for the drive
16159                                   ;        bp    - contains retry count
16160                                   ; on exit    flags indicate result of decrementing retry count
16161
16162                                   again:
16163 00000DE1 E85705                             call    ResetDisk
16164 00000DE4 80FC06                             cmp     ah, 6
16165 00000DE7 7402                               jz      short dont_dec_retry_count ; If      it is a media change error
16166                                                                                ; do not decrement retry count.
16167 00000DE9 4D                                 dec     bp              ; decrement retry count
16168 00000DEA C3                                 retn
16169                                   ; ---------------------------------------------------------------------------
16170
16171                                   dont_dec_retry_count:
16172 00000DEB 08E4                               or      ah, ah
16173 00000DED C3                                 retn
16174
16175                                   ;---------------------------------------------------------------------------
16176                                   ; Retro DOS v5.0 - PCDOS 7.1 IBMBIO.COM - BIOSCODE:0E30h
16177                                   ;---------------------------------------------------------------------------
16178                                   ; 24/12/2023 - Retro DOS v5.0
16179                                   ;;;;
16180
16181 00000DEE 00                       ioctl_drvnum:       db 0
16182
16183                                             ; 24/12/2023
16184
16185                                   ; =============== S U B R O U T I N E =======================================
16186
16187                                   get_phy_drv_num:
16188 00000DEF E8B2F7                             call    SetDrive        ; get physical drive number
16189                                                                     ; INPUT: al = logical drive number (BDS.drivelet)
16190                                                                     ; OUTPUT: physical drive number (BDS.drivenum)
16191 00000DF2 268A5504                           mov     dl, [es:di+4]   ; [es:di+BDS.drivenum]
16192 00000DF6 C3                                 retn
16193
16194                                   ; =============== S U B R O U T I N E =======================================
16195
16196                                             ; 24/12/2023
16197                                   ioctl_output:
16198 00000DF7 E8F5FF                             call    get_phy_drv_num
16199 00000DFA 2E8816[EE0D]                       mov     [cs:ioctl_drvnum], dl
16200 00000DFF B441                               mov     ah, 41h
16201 00000E01 BBAA55                             mov     bx, 55AAh
16202 00000E04 CD13                               int     13h             ; DISK - Check for INT 13h Extensions
16203                                                                     ; BX = 55AAh, DL = drive number
16204                                                                     ; Return: CF set if not supported
16205                                                                     ; AH = extensions version
16206                                                                     ; BX = AA55h
16207                                                                     ; CX = Interface support bit map
16208 00000E06 7235                               jc      short int13h_exts_err
16209                                   ioctl_input_1:
16210 00000E08 C43E[1200]                         les     di, [ptrsav]
16211 00000E0C 26C47D0E                           les     di, [es:di+14] ; [es:di+IOCTL_REQ.MINORFUNCTION]
16212 00000E10 723E                               jc      short ioctl_input_2
16213 00000E12 B80046                             mov     ax, 4600h       ; Eject removable media
16214 00000E15 263805                             cmp     [es:di], al     ; al = 0 ; disk ioctl function = 0
16215 00000E18 7417                               je      short ioctl_output_1
16216 00000E1A 26803D01                           cmp     byte [es:di], 1 ; al = 1 ; disk ioctl function = 1
16217 00000E1E 751B                               jne     short ioctl_output_2
16218 00000E20 B80145                             mov     ax, 4501h       ; Lock/unlock media
16219                                                                     ; (al, 0 = lock, 1 = unlock)
16220 00000E23 26807D0100                         cmp     byte [es:di+1], 0 ; unlock (reverse of INT 13h ah=45h)
16221 00000E28 7407                               jz      short ioctl_output_1
16222 00000E2A 26384501                           cmp     [es:di+1], al   ; lock (reverse of INT 13h ah=45h)
16223 00000E2E 750B                               jne     short ioctl_output_2
16224 00000E30 48                                 dec     ax
16225                                   ioctl_output_1:
16226 00000E31 2E8A16[EE0D]                       mov     dl, [cs:ioctl_drvnum]
16227 00000E36 CD13                               int     13h             ; DISK - IBM/MS Extension - LOCK/UNLOCK DRIVE (DL - drive, [SI - disk address
packet)
16228 00000E38 7203                               jc      short int13h_exts_err
16229                                   ioctl_lock_err:
16230                                             ; cf=0
16231                                   ioctl_output_ret:
16232                                             ;clc
16233 00000E3A C3                                 retn
16234                                   ; ---------------------------------------------------------------------------
16235
16236                                   ioctl_output_2:
16237 00000E3B B401                               mov     ah, 1
16238                                   int13h_exts_err:
16239 00000E3D 80FCB0                             cmp     ah, 0B0h        ; volume not locked in drive
16240 00000E40 74F8                               je      short ioctl_lock_err
16241 00000E42 80FCB4                             cmp     ah, 0B4h        ; lock count exceeded
16242 00000E45 74F3                               je      short ioctl_lock_err
16243 00000E47 E9DAF7                             jmp     err_exitj
16244
```

```
                                ; ============== 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
```

```
16368 00000EA2 [160F]                          dw Cmd_Error_Proc
16369 00000EA4 [160F]                          dw Cmd_Error_Proc       ; 6Eh
16370 00000EA6 [C815]                          dw GetDrvMapInfo        ; 6Fh
16371
16372                                  IoWriteJumpTable:
16373 00000EA8 0A                              db ((IOC_DC_Table-IoWriteJumpTable)-1)/2 ; 9
16374 00000EA9 [7A0F]                          dw SetDeviceParameters; 40h
16375 00000EAB [9A11]                          dw WriteTrack           ; 41h
16376 00000EAD [6D10]                          dw FormatTrack          ; 42h
16377 00000EAF [160F]                          dw Cmd_Error_Proc
16378 00000EB1 [160F]                          dw Cmd_Error_Proc
16379 00000EB3 [160F]                          dw Cmd_Error_Proc
16380 00000EB5 [5214]                          dw SetMediaId           ; 46h
16381 00000EB7 [0415]                          dw SetAccessFlag        ; 47h ; unpublished function
16382                                          ; 24/12/2023
16383                                          ; Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
16384 00000EB9 [8115]                          dw SetLockState                 ; 48h
16385 00000EBB [9815]                          dw EjectMedia           ; 49h
16386
16387                                  ; ============================================================================
16388                                  ; IOC_DC_Table
16389                                  ;
16390                                  ; This table contains all of the valid generic IOCtl Minor codes for
16391                                  ; major function 08 to be used by the Ioctl_Support_Query function.
16392                                  ; Added for 5.00
16393                                  ; ============================================================================
16394
16395                                          ; 24/12/2023 - Retro DOS v5.0
16396                                          ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0F00h)
16397
16398 00000EBD 60                     IOC_DC_Table:      db 60h                ; GET_DEVICE_PARAMETERS
16399 00000EBE 40                              db 40h                  ; SET_DEVICE_PARAMETERS
16400 00000EBF 61                              db 61h                  ; READ_TRACK
16401 00000EC0 41                              db 41h                  ; WRITE_TRACK
16402 00000EC1 62                              db 62h                  ; VERIFY_TRACK
16403 00000EC2 42                              db 42h                  ; FORMAT_TRACK
16404 00000EC3 66                              db 66h                  ; GET_MEDIA_ID
16405 00000EC4 46                              db 46h                  ; SET_MEDIA_ID
16406 00000EC5 67                              db 67h                  ; GET_ACCESS_FLAG
16407 00000EC6 47                              db 47h                  ; SET_ACCESS_FLAG
16408 00000EC7 68                              db 68h                  ; SENSE_MEDIA_TYPE
16409                                          ; 24/12/2023
16410                                          ; Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
16411 00000EC8 48                              db 48h                  ; SET_LOCK_STATE
16412 00000EC9 49                              db 49h                  ; EJECT_MEDIA
16413 00000ECA 6F                              db 6Fh                  ; GET_DRV_MAP_INFO
16414
16415                                  ;IOC_DC_TABLE_LEN EQU $ - IOC_DC_Table
16416
16417                                          ; 24/12/2023 - Retro DOS v5.0
16418                                          ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0F0Eh)
16419
16420 00000ECB 00                     new_genioctl:      db 0
16421
16422                                  ; ----------------------------------------------------------------------------
16423
16424                                  ; 16/10/2022
16425
16426                                  ; ============================================================================
16427                                  ; Do_Generic_IOCtl: perform generic ioctl request
16428                                  ;
16429                                  ;    input: AL contains logical drive
16430                                  ;
16431                                  ;   functions are dispatched through a call. On return, carry indicates
16432                                  ;   error code in al. Note::bES:b& ds undefined on return from
16433                                  ;   subfunctions.
16434                                  ;
16435                                  ; ============================================================================
16436
16437                                  ; 11/03/2019
16438                                          ; 24/12/2023 - Retro DOS v5.0
16439                                          ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0F0Fh)
16440
16441                                          ; 19/10/2022
16442                                  do_generic_ioctl:                 ; 2C7h:0C6Bh = 70h:31DBh
16443 00000ECC E8D5F6                          call   SetDrive         ; ES:DI Points to bds for drive
16444
16445                                          ; 24/12/2023
16446                                          ;;;
16447 00000ECF 2EC606[CB0E]00                  mov    byte [cs:new_genioctl], 0
16448                                                                  ; 0, old generic ioctl function
16449 00000ED5 06                              push   es
16450 00000ED6 C41E[1200]                      les    bx, [ptrsav]    ; ES:BX Points to request header
16451 00000EDA 26807F0D08                      cmp    byte [es:bx+0Dh], 8 ; [es:bx+IOCTL_REQ.MAJORFUNCTION]
16452                                                                  ; RAWIO
16453                                          ; 24/12/2023
16454                                          ;mov    al, [es:bx+0Eh]        ; [es:bx+IOCTL_REQ.MINORFUNCTION]
16455                                          ;pop    es
16456                                          ;jnz    short IoctlFuncErr
16457 00000EDF 740A                            jz     short chk_genioctl_minor
16458 00000EE1 2EFE06[CB0E]                    inc    byte [cs:new_genioctl]
16459                                                                  ; 1, new generic ioctl function (FAT32)
16460 00000EE6 26807F0D48                      cmp    byte [es:bx+0Dh], 48h ; Generic IOCtl Request support
16461                                                                  ; (called only if bit 6 of attribute is set to 1)
16462                                  chk_genioctl_minor:
16463 00000EEB 268A470E                        mov    al, [es:bx+0Eh]        ; [es:bx+IOCTL_REQ.MINORFUNCTION]
16464 00000EEF 07                              pop    es
16465 00000EF0 7525                            jnz    short IoctlFuncErr
16466                                          ;;;
16467
16468                                          ; cas note: Could do the above two blocks in reverse order.
16469                                          ; Would have to preserve al for SetDrive
16470
16471                                          ; 10/12/2022
16472 00000EF2 BE[870E]                        mov    si, IoReadJumpTable
16473                                          ;mov    si, 0C3Ch      ; IoReadJumpTable
16474                                                                  ; at 2C7h:0C3Ch       = 70h:31ACh
16475 00000EF5 A820                            test   al, 20h        ; GEN_IOCTL_FN_TST ; test of req. function
16476 00000EF7 7503                            jnz    short NotGenericWrite ; function is a read.
16477                                          ; 10/12/2022
16478 00000EF9 BE[A80E]                        mov    si, IoWriteJumpTable
16479                                          ;mov    si, 0C4Fh      ; IoWriteJumpTable
16480                                                                  ; at 2C7h:0C4Fh       = 70h:31BFh
16481                                  NotGenericWrite:
16482 00000EFC 24DF                            and    al, 0DFh        ; ~GEN_IOCTL_FN_TST ; get rid of read/write bit
16483 00000EFE 2C40                            sub    al, 40h         ; offset for base function
16484 00000F00 2E3A04                          cmp    al, [cs:si]
16485 00000F03 7712                            ja     short IoctlFuncErr
16486 00000F05 98                              cbw
16487                                          ; 24/12/2023
16488                                          ;shl    ax, 1
16489 00000F06 01C0                            add    ax, ax
16490 00000F08 46                              inc    si
16491 00000F09 01C6                            add    si, ax
```

```
16492 00000F0B 2EFF14                           call    near [cs:si]
16493                                           ;call   word ptr cs:[si]
16494 00000F0E 2E8E1E[3000]                     mov     ds, [cs:BIOSDATAWORD]
16495                                           ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16496                                                                   ; 2C7h:30h = 70h:25A0h
16497 00000F13 B481                             mov     ah, 81h         ; Return this status in       case of carry
16498 00000F15 C3                               retn                    ; Pass carry flag through to exit code
16499                                   ; ---------------------------------------------------------------------------
16500
16501                                           ; Cmd_Error_Proc is called as a procedure and also use
16502                                           ; as a fall through from above
16503                                   Cmd_Error_Proc:                         ; 2C7h:0CA3h = 70h:3213h
16504 00000F16 5A                               pop     dx
16505                                   IoctlFuncErr:
16506 00000F17 E9BBF1                           jmp     bc_cmderr
16507                                   ; ---------------------------------------------------------------------------
16508
16509                                   ; 16/10/2022
16510
16511                                   ; ============================================================================
16512                                   ;** GetDeviceParameters:
16513                                   ;
16514                                   ;   GetDeviceParameters implements the generic ioctl function:
16515                                   ;   majorcode=RAWIO, minorcode=GetDeviceParameters (60h)
16516                                   ;
16517                                   ;   ENTRY   (ES:di) = BDS for drive
16518                                   ;           PtrSav = long pointer to request header
16519                                   ;   EXIT    ??? BUGBUG
16520                                   ;   USES    ??? BUGBUG
16521                                   ; ============================================================================
16522
16523                                           ; 24/12/2023 - Retro DOS v5.0
16524                                           ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0F5Dh)
16525
16526                                           ; 19/10/2022
16527                                   GetDeviceParameters:
16528                                           ; Copy info from bds to the device parameters packet
16529
16530 00000F1A C51E[1200]                       lds     bx, [ptrsav]    ; DS:BX points to request header
16531 00000F1E C55F13                           lds     bx, [bx+19]     ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET]
16532                                                                   ; (DS:BX) = return buffer
16533                                           ; 24/12/2023
16534 00000F21 268A453E                         mov     al, [es:di+3Eh]
16535                                           ;mov    al, [es:di+34] ; [es:di+BDS.formfactor]
16536 00000F25 884701                           mov     [bx+1], al      ; [bx+A_DEVICEPARAMETERS.DP_DEVICETYPE]
16537                                           ; 24/12/2023
16538 00000F28 268B453F                         mov     ax, [es:di+3Fh]
16539                                           ;mov    ax, [es:di+35] ; [es:di+BDS.flags]
16540 00000F2C 83E003                           and     ax, 3           ; fnon_removable+fchangeline
16541                                                                   ; Mask off other bits
16542 00000F2F 894702                           mov     [bx+2], ax      ; [bx+A_DEVICEPARAMETERS.DP_DEVICEATTRIBUTES]
16543                                           ; 24/12/2023
16544 00000F32 268B4541                         mov     ax, [es:di+41h]
16545                                           ;mov    ax, [es:di+37] ; [es:di+BDS.cylinders]
16546 00000F36 894704                           mov     [bx+4], ax      ; [bx+A_DEVICEPARAMETERS.DP_CYLINDERS]
16547 00000F39 30C0                             xor     al, al          ; Set media type to default
16548 00000F3B 884706                           mov     [bx+6], al      ; [bx+A_DEVICEPARAMETERS.DP_MEDIATYPE]
16549
16550                                           ; copy recommended bpb
16551
16552                                           ; 24/12/2023
16553 00000F3E 8D7543                           lea     si, [di+43h]
16554                                           ;lea    si, [di+39]     ; [di+BDS.rbytespersec]       = [di+BDS.R_BPB]
16555 00000F41 F60701                           test    byte [bx], 1    ; [bx+A_DEVICEPARAMETERS.DP_SPECIALFUNCTIONS]
16556                                                                   ; BUILD_DEVICE_BPB
16557 00000F44 7412                             jz      short UseBpbPresent
16558 00000F46 1E                               push    ds              ; Save request packet segment
16559 00000F47 2E8E1E[3000]                     mov     ds, [cs:BIOSDATAWORD]
16560                                           ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16561                                                                   ; 2C7h:30h = 70h:25A0h
16562                                                                   ; Point back to Bios_Data
16563 00000F4C E828FA                           call    checksingle
16564 00000F4F E884F7                           call    GetBp           ; Build the bpb from scratch
16565 00000F52 1F                               pop     ds              ; Restore request packet segment
16566 00000F53 7224                             jb      short GetParmRet
16567 00000F55 8D7506                           lea     si, [di+6]      ; [di+BDS.bytespersec] = [di+BSD.DP_BPB]
16568                                                                   ; Use this subfield of bds instead
16569                                   UseBpbPresent:
16570 00000F58 8D7F07                           lea     di, [bx+7]      ; [bx+A_DEVICEPARAMETERS.DP_BPB]
16571                                                                   ; This is where       the result goes
16572                                           ; 24/12/2023
16573 00000F5B 31D2                             xor     dx, dx ; 0
16574
16575                                           ; 24/12/2023
16576 00000F5D B91F00                           mov     cx, 31          ; A_BPB.size = 31
16577                                           ;mov    cx, 25          ; A_BPB.size - 6
16578                                                                   ; For now use 'small' bpb
16579                                           ; 24/12/2023
16580                                           ;;;
16581 00000F60 2E3816[CB0E]                     cmp     [cs:new_genioctl], dl ; 0 ? ; *
16582 00000F65 7404                             jz      short gdp_1     ; old type (FAT12 & FAT16) structure
16583                                           ;mov    cx, 53          ; FAT32 BPB size
16584                                           ;mov    dx, 32          ; 53+32 = 85 bytes (A_BPB_FAT32.size)
16585 00000F67 B135                             mov     cl, 53
16586 00000F69 B220                             mov     dl, 32
16587                                   gdp_1:
16588                                           ;;;
16589 00000F6B 1E                               push    ds              ; reverse segments for copy
16590 00000F6C 06                               push    es
16591 00000F6D 1F                               pop     ds
16592 00000F6E 07                               pop     es
16593 00000F6F F3A4                             rep movsb
16594
16595                                           ; 24/12/2023
16596                                           ;;;
16597 00000F71 89D1                             mov     cx, dx          ; 0 or 32
16598 00000F73 E304                             jcxz    gdp_2
16599 00000F75 30C0                             xor     al, al          ; 32 zeros
16600 00000F77 F3AA                             rep stosb
16601                                   gdp_2:
16602                                           ;clc    ; cf is already 0 ; * ; 24/12/2023
16603                                           ;;;
16604
16605                                           ; 12/12/2022
16606                                           ; cf=0 (cmp instruction -above- resets cf)
16607                                           ;clc
16608                                   GetParmRet:
16609 00000F79 C3                               retn
16610                                   ; ---------------------------------------------------------------------------
16611
16612                                   ; 17/10/2022
16613                                   ; 16/10/2022
16614
16615                                   ; ============================================================================
```

```
16616                                   ; SetDeviceParameters:
16617                                   ;
16618                                   ; input: ES:di points to bds for drive
16619                                   ; ================================================================================
16620
16621                                           ; 24/12/2023 - Retro DOS v5.0
16622                                           ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:0FC0h)
16623
16624                                           ; 19/10/2022
16625                                   SetDeviceParameters:                    ; 2C7h:0CF3h = 70h:3263h
16626 00000F7A C51E[1200]                       lds     bx, [ptrsav]    ; DS:BX points to request header
16627 00000F7E C55F13                           lds     bx, [bx+19]     ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET]
16628                                           ; 24/12/2023
16629 00000F81 26814D3F4001                     or      word [es:di+3Fh], 140h
16630                                           ;or     word [es:di+23h], 140h ; [es:di+BDS.flags]
16631                                                                   ; fchanged_by_format|fchanged
16632 00000F87 F60702                           test    byte [bx], 2    ; [bx+A_DEVICEPARAMETERS.DP_SPECIALFUNCTIONS]
16633                                                                   ; ONLY_SET_TRACKLAYOUT
16634                                           ;jnz    short setTrackTable
16635                                           ; 24/12/2023
16636 00000F8A 7403                             jz      short sdp_1
16637 00000F8C E98000                           jmp     setTrackTable
16638                                   sdp_1:
16639 00000F8F 8A4701                           mov     al, [bx+1]      ; [bx+A_DEVICEPARAMETERS.DP_DEVICETYPE]
16640                                           ; 24/12/2023
16641 00000F92 2688453E                         mov     [es:di+3Eh], al
16642                                           ;mov    [es:di+34], al ; [es:di+BDS.formfactor]
16643 00000F96 8B4704                           mov     ax, [bx+4]      ; [bx+A_DEVICEPARAMETERS.DP_CYLINDERS]
16644                                           ; 24/12/2023
16645 00000F99 26894541                         mov     [es:di+41h], ax
16646                                           ;mov    [es:di+37], ax ; [es:di+BDS.cylinders]
16647 00000F9D 8B4702                           mov     ax, [bx+2]      ; [bx+A_DEVICEPARAMETERS.DP_DEVICEATTRIBUTES]
16648 00000FA0 1E                               push    ds
16649                                           ; 17/10/2022
16650 00000FA1 2E8E1E[3000]                     mov     ds, [cs:BIOSDATAWORD]
16651                                           ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16652                                                                   ; 2C7h:30h = 70h:25A0h
16653                                           ;cmp    byte [fhave96], 0
16654 00000FA6 803E[7700]00                     cmp     byte [fhave96], 0
16655 00000FAB 1F                               pop     ds
16656 00000FAC 7502                             jnz     short HaveChange ; we have changeline support
16657                                           ; 10/12/2022
16658 00000FAE 24FD                             and     al, 0FDh
16659                                           ;and    ax, 0FFFDh      ; ~fchangeline
16660
16661                                           ; Ignore all bits except non_removable and changeline
16662                                   HaveChange:
16663 00000FB0 83E003                           and     ax, 3           ; fnon_removable|fchangeline
16664                                           ; 24/12/2023
16665 00000FB3 268B4D3F                         mov     cx, [es:di+3Fh]
16666                                           ;mov    cx, [es:di+35] ; [es:di+BDS.flags]
16667 00000FB7 81E1F4FD                         and     cx, 0FDF4h      ; ~(fnon_removable|fchangeline|good_tracklayout|unformatted_media)
16668 00000FBB 09C8                             or      ax, cx
16669                                           ; 24/12/2023
16670 00000FBD 2689453F                         mov     [es:di+3Fh], ax
16671                                           ;mov    [es:di+35], ax ; [es:di+BDS.flags]
16672 00000FC1 8A4706                           mov     al, [bx+6]      ; [bx+A_DEVICEPARAMETERS.DP_MEDIATYPE]
16673                                                                   ; Set media type
16674 00000FC4 1E                               push    ds
16675 00000FC5 2E8E1E[3000]                     mov     ds, [cs:BIOSDATAWORD]
16676                                           ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16677 00000FCA A2[A805]                         mov     [mediatype], al
16678                                           ;mov    ds:mediatype, al
16679
16680                                           ; 24/12/2023
16681                                           ;;;
16682 00000FCD B93500                           mov     cx, 53          ; FAT32 BPB size
16683 00000FD0 2E803E[CB0E]00                   cmp     byte [cs:new_genioctl], 0
16684 00000FD6 7502                             jnz     short sdp_2     ; new type (FAT32) structure
16685                                           ;mov    cx, 31          ; A_BPB.size = 31
16686 00000FD8 B11F                             mov     cl, 31
16687                                   sdp_2:
16688                                           ;;;
16689 00000FDA 1F                               pop     ds
16690
16691                                           ; The media changed (maybe) so we will have to do a set dasd
16692                                           ; the next time we format a track
16693
16694                                           ; 24/12/2023
16695 00000FDB 26804D3F80                       or      byte [es:di+3Fh], 80h
16696                                           ; 10/12/2022
16697                                           ;or     byte [es:di+35], 80h
16698                                           ;;or    word [es:di+35], 80h ; [es:di+BDS.flags]
16699                                                                   ; set_dasd_true
16700 00000FE0 57                               push    di              ; Save bds pointer
16701
16702                                           ; Figure out what we are supposed to do with the bpb
16703                                           ; were we asked to install a fake bpb?
16704
16705 00000FE1 F60701                           test    byte [bx], 1    ; [bx+A_DEVICEPARAMETERS.DP_SPECIALFUNCTIONS]
16706                                                                   ; INSTALL_FAKE_BPB
16707 00000FE4 7511                             jnz     short InstallFakeBpb
16708
16709                                           ; were we returning a fake bpb when asked to build a bpb?
16710
16711                                           ; 24/12/2023
16712 00000FE6 26F6453F04                       test    byte [es:di+3Fh], 4
16713                                           ; 10/12/2022
16714                                           ;test   byte [es:di+35], 4
16715                                           ;;test  word [es:di+35], 4 ; [es:di+BDS.flags]
16716                                                                   ; return_fake_bpb
16717 00000FEB 7405                             jz      short InstallRecommendedBpb
16718
16719                                           ; we were returning a fake bpb but we can stop now
16720
16721                                           ; 24/12/2023
16722 00000FED 2680653FFB                       and     byte [es:di+3Fh], 0FBh
16723                                           ; 10/12/2022
16724                                           ;and    byte [es:di+35], 0FBh
16725                                           ;;and   word [es:di+35], 0FFFBh ; [es:di+BDS.flags]
16726                                                                   ; ~return_fake_bpb
16727                                   InstallRecommendedBpb:
16728                                           ; 24/12/2023
16729                                           ;mov    cx, 31          ; A_BPB.size
16730                                           ;lea    di, [di+27h]    ; [di+BDS.R_BPB] = [di+BDS.rbytespersec]
16731                                           ; cx = 53 or 31
16732 00000FF2 8D7D43                           lea     di, [di+43h]    ; (PCDOS 7.1 IBMBIO.COM)
16733 00000FF5 EB08                             jmp     short CopyTheBpb
16734                                   ; ---------------------------------------------------------------------------
16735
16736                                   InstallFakeBpb:
16737                                           ; 24/12/2023
16738 00000FF7 26804D3F04                       or      byte [es:di+3Fh], 4
16739                                           ; 10/12/2022
```

```
16740                                           ;or     byte [es:di+35], 4
16741                                           ;;or    word [es:di+35], 4 ; byte [es:di+BDS.flags]
16742                                                   ; return_fake_bpb
16743                                           ; 24/12/2023
16744                                           ; cx = 53 or 31
16745                                           ;mov    cx, 25          ; A_BPB.size - 6
16746                                                   ; move 'smaller' bpb
16747 00000FFC 8D7D06                           lea     di, [di+6]      ; [es:di+BDS.BPB] = [es:di+BDS.bytespersec]
16748                                   CopyTheBpb:
16749 00000FFF 8D7707                           lea     si, [bx+7]      ; [bx+A_DEVICEPARAMETERS.DP_BPB]
16750 00001002 F3A4                             rep movsb
16751 00001004 1E                               push    ds              ; Save packet segment
16752                                           ; 17/10/2022
16753 00001005 2E8E1E[3000]                     mov     ds, [cs:BIOSDATAWORD]
16754                                           ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16755                                                   ; Setup for ds -> Bios_Data
16756 0000100A E8DD03                           call    RestoreOldDpt   ; Restore the old Dpt from TempDpt
16757 0000100D 1F                               pop     ds              ; Restore packet segment
16758 0000100E 5F                               pop     di              ; Restore bds pointer
16759                                   setTrackTable:
16760                                           ; 24/12/2023
16761                                           ;mov    cx, [bx+38]     ; [bx+26h]
16762                                           ;;;
16763 0000100F 8B4F5C                           mov     cx, [bx+5Ch]    ; [bx+A_DEVICEPARAMETERS.DP_TRACKTABLEENTRIES]
16764                                                   ; offset 85+7 (A_BPB.size+7) (FAT32)
16765 00001012 2E803E[CB0E]00                   cmp     byte [cs:new_genioctl], 0
16766 00001018 7503                             jnz     short sdp_3     ; new type (FAT32) structure
16767 0000101A 8B4F26                           mov     cx, [bx+26h]    ; [bx+A_DEVICEPARAMETERS.DP_TRACKTABLEENTRIES]
16768                                                   ; offset 31+7 (A_BPB.size+7)
16769                                   sdp_3:
16770                                           ;;;
16771
16772 0000101D 1E                               push    ds
16773 0000101E 2E8E1E[3000]                     mov     ds, [cs:BIOSDATAWORD]
16774 00001023 890E[AA04]                       mov     [sectorspertrack], cx
16775 00001027 1F                               pop     ds
16776
16777                                           ; 24/12/2023
16778 00001028 2680653FF7                       and     byte [es:di+3Fh], 0F7h
16779                                           ; 10/12/2022
16780                                           ;and    byte [es:di+35], 0F7h
16781                                           ;;and   word [es:di+35], 0FFF7h ; [es:di+BDS.flags]
16782                                                   ; ~good_tracklayout
16783 0000102D F60704                           test    byte [bx], 4    ; [bx+A_DEVICEPARAMETERS.DP_SPECIALFUNCTIONS]
16784                                                   ; TRACKLAYOUT_IS_GOOD
16785 00001030 7405                             jz      short UglyTrackLayOut
16786                                           ; 24/12/2023
16787 00001032 26804D3F08                       or      byte [es:di+3Fh], 8
16788                                           ; 10/12/2022
16789                                           ;or     byte [es:di+35], 8
16790                                           ;;or    word [es:di+35], 8 ; [es:di+BDS.flags]
16791                                                   ; good_tracklayout
16792                                   UglyTrackLayOut:
16793 00001037 83F93F                           cmp     cx, 63          ; MAX_SECTORS_IN_TRACK
16794 0000103A 772D                             ja      short TooManyPerTrack
16795                                           ;jcxz   short SectorInfoSaved
16796 0000103C E329                             jcxz    SectorInfoSaved       ; 19/10/2022
16797
16798 0000103E BF[AC04]                         mov     di, tracktable
16799
16800                                           ; 24/12/2023
16801                                           ;lea    si, [bx+40]     ; [bx+A_DEVICEPARAMETERS.DP_SECTORTABLE]
16802                                           ;;;
16803 00001041 8D775E                           lea     si, [bx+5Eh]    ; [bx+A_DEVICEPARAMETERS.DP_SECTORTABLE]
16804                                                   ; offset 85+9 (A_BPB.size+9) (FAT32)
16805 00001044 2E803E[CB0E]00                   cmp     byte [cs:new_genioctl], 0
16806 0000104A 7503                             jnz     short sdp_4     ; new type (FAT32) structure
16807 0000104C 8D7728                           lea     si, [bx+28h]    ; [bx+A_DEVICEPARAMETERS.DP_SECTORTABLE]
16808                                                   ; offset 31+9 (A_BPB.size+9)
16809                                   sdp_4:
16810                                           ;;;
16811
16812                                           ; 17/10/2022
16813 0000104F 2E8E06[3000]                     mov     es, [cs:BIOSDATAWORD]
16814                                           ;mov    es, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16815                                                   ; Trash our bds pointer
16816                                   StoreSectorInfo:
16817 00001054 47                               inc     di
16818 00001055 47                               inc     di              ; Skip over cylinder and head
16819 00001056 AD                               lodsw                   ; Get sector id
16820 00001057 AA                               stosb                   ; Copy it
16821 00001058 AD                               lodsw                   ; Get sector size
16822
16823                                           ; 24/12/2023
16824                                           ; 02/09/2023 (PCDOS 7.1)
16825                                           ;call   SectSizeToSectIndex
16826                                           ; 18/04/2024
16827                                           ;cmp    ah, 3 ; 02/09/2023
16828 00001059 80FC02                           cmp     ah, 2           ; (0=>128,1=>256,2=>512,3=>1024)
16829                                                   ; examine upper       byte only
16830 0000105C 7704                             ja      short OneK
16831 0000105E 88E0                             mov     al, ah          ; value in AH is the index!
16832 00001060 EB02                             jmp     short sdp_s
16833                                   OneK:
16834 00001062 B003                             mov     al, 3           ; 1024 bytes per sector
16835                                   sdp_s:
16836 00001064 AA                               stosb                   ; Store sector SIZE index
16837 00001065 E2ED                             loop    StoreSectorInfo
16838                                   SectorInfoSaved:
16839 00001067 F8                               clc
16840 00001068 C3                               retn
16841                                   ; ---------------------------------------------------------------------------
16842
16843                                   TooManyPerTrack:
16844 00001069 B00C                             mov     al, 0Ch
16845 0000106B F9                               stc
16846 0000106C C3                               retn
16847                                   ; ---------------------------------------------------------------------------
16848
16849                                   ; 16/10/2022
16850
16851                                   ; ============================================================================
16852                                   ; FormatTrack:
16853                                   ; if specialfunction byte is 1,then this is a status call to see if there is
16854                                   ; rom support for the combination of sec/trk and # of cyln,and if the
16855                                   ; combination is legal. if specialfunction byte is 0,then format the track.
16856                                   ;
16857                                   ; input: ES:di points to bds for drive
16858                                   ;
16859                                   ; output:
16860                                   ;   for status call:
16861                                   ;   specialfunction byte set to:
16862                                   ;           0 - rom support + legal combination
16863                                   ;           1 - no rom support
```

```
16864                                   ;           2 - illegal combination
16865                                   ;           3 - no media present
16866                                   ;   carry cleared.
16867                                   ;
16868                                   ;   for format track:
16869                                   ;           carry set if error
16870                                   ;
16871                                   ; ====================================================================
16872
16873                                   ; 16/03/2019
16874                                               ; 24/12/2023 - Retro DOS 5.0
16875                                               ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:10B7h)
16876
16877                                               ; 19/10/2022
16878                                   FormatTrack:
16879 0000106D C51E[1200]                           lds     bx, [ptrsav]
16880 00001071 C55F13                               lds     bx, [bx+19]     ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET
16881 00001074 F60701                               test    byte [bx], 1    ; [bx+A_DEVICEPARAMETERS.DP_SPECIALFUNCTIONS]
16882                                                                       ; STATUS_FOR_FORMAT
16883 00001077 740E                                 jz      short DoFormatTrack
16884 00001079 1E                                   push    ds
16885                                               ; 17/10/2022
16886 0000107A 2E8E1E[3000]                         mov     ds, [cs:BIOSDATAWORD]
16887                                               ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16888 0000107F E82502                               call    SetMediaForFormat ; Also moves current Dpt to TempDpt
16889 00001082 1F                                   pop     ds
16890 00001083 8807                                 mov     [bx], al        ; [bx+A_DEVICEPARAMETERS.DP_SPECIALFUNCTIONS]
16891 00001085 F8                                   clc
16892 00001086 C3                                   retn
16893                                   ; ---------------------------------------------------------------------------
16894
16895                                   DoFormatTrack:
16896                                               ; 24/12/2023 - Retro DOS 5.0
16897 00001087 26807D3E05                           cmp     byte [es:di+3Eh], 5
16898                                               ;cmp    byte [es:di+34], 5 ; [es:di+BDS.formfactor]
16899                                                                       ; DEV_HARDDISK
16900 0000108C 7508                                 jnz     short DoFormatDiskette
16901                                               ; 17/10/2022
16902 0000108E 2E8E1E[3000]                         mov     ds, [cs:BIOSDATAWORD]
16903                                               ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16904                                                                       ; Point to Bios_Data (at 2C7h:30h or 70h:25A0h)
16905 00001093 E99D00                               jmp     VerifyTrack
16906                                   ; ---------------------------------------------------------------------------
16907
16908                                   DoFormatDiskette:
16909 00001096 8B4F01                               mov     cx, [bx+1]
16910 00001099 8B5703                               mov     dx, [bx+3]
16911 0000109C F60702                               test    byte [bx], 2
16912                                               ; 17/10/2022
16913 0000109F 2E8E1E[3000]                         mov     ds, [cs:BIOSDATAWORD]
16914                                               ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
16915                                                                       ; Setup ds-> Bios_Data for verify
16916 000010A4 7403                                 jz      short DoFormatDiskette_1
16917 000010A6 E9E500                               jmp     VerifyTrack_Err
16918                                   ; ---------------------------------------------------------------------------
16919
16920                                   DoFormatDiskette_1:
16921 000010A9 E8FB01                               call    SetMediaForFormat ; Also moves current Dpt to TempDpt
16922 000010AC 3C01                                 cmp     al, 1           ;  ROM support for sec/trk,# trks comb?
16923 000010AE 7406                                 jz      short NeedToSetDasd ; Old rom
16924 000010B0 3C03                                 cmp     al, 3           ; Time out error?
16925 000010B2 7507                                 jnz     short NoSetDasd       ; No,fine. (at this point, don't care
16926                                                                       ; about the illegal combination)
16927 000010B4 EB68                                 jmp     short FormatFailed
16928                                   ; ---------------------------------------------------------------------------
16929
16930                                   NeedToSetDasd:
16931 000010B6 52                                   push    dx
16932 000010B7 E89001                               call    SetDasd         ; INT 13h, AH=17h
16933 000010BA 5A                                   pop     dx
16934                                   NoSetDasd:
16935 000010BB E8B9F8                               call    checksingle     ; Do any needed       diskette swapping
16936 000010BE 89D0                                 mov     ax, dx          ; Get track from packet
16937 000010C0 A3[3901]                             mov     [trknum], ax
16938 000010C3 880E[3801]                           mov     [hdnum], cl     ; Store head from packet
16939 000010C7 88CC                                 mov     ah, cl
16940 000010C9 BB[AC04]                             mov     bx, tracktable
16941 000010CC 8B0E[AA04]                           mov     cx, [sectorspertrack]
16942                                               ; 24/12/2023 - Retro DOS 5.0
16943 000010D0 E307                                 jcxz    set_fmt_retry_count
16944                                   StoreCylinderHead:
16945 000010D2 8907                                 mov     [bx], ax        ; Store into TrackTable
16946 000010D4 83C304                               add     bx, 4           ; Skip to next sector field
16947 000010D7 E2F9                                 loop    StoreCylinderHead
16948                                   set_fmt_retry_count:        ; 24/12/2023
16949                                               ;mov    cx, 5           ; MAXERR - Set up retry       count
16950                                               ; 02/09/2023
16951 000010D9 B105                                 mov     cl, 5
16952                                   FormatRetry:
16953 000010DB 51                                   push    cx
16954 000010DC BB[AC04]                             mov     bx, tracktable
16955 000010DF A0[AA04]                             mov     al, [sectorspertrack]
16956 000010E2 B405                                 mov     ah, 5           ; romformat
16957 000010E4 8C1E[A804]                           mov     [xfer_seg], ds
16958 000010E8 E86602                               call    ToRom
16959 000010EB 59                                   pop     cx
16960 000010EC 7216                                 jb      short FormatError
16961 000010EE 51                                   push    cx              ; Now verify the sectors just formatted.
16962                                                                       ; NOTE: because of bug in some BIOSes we have to
16963                                                                       ;       set ES:BX to 00:00
16964 000010EF 53                                   push    bx
16965 000010F0 31DB                                 xor     bx, bx
16966 000010F2 891E[A804]                           mov     [xfer_seg], bx
16967 000010F6 A0[AA04]                             mov     al, [sectorspertrack]
16968 000010F9 B404                                 mov     ah, 4           ; romverify
16969 000010FB B101                                 mov     cl, 1
16970 000010FD E85102                               call    ToRom
16971 00001100 5B                                   pop     bx
16972 00001101 59                                   pop     cx
16973 00001102 7329                                 jnb     short FormatOk
16974                                   FormatError:
16975 00001104 E83402                               call    ResetDisk
16976 00001107 C606[AA05]01                         mov     byte [had_format_error], 1
16977 0000110C 50                                   push    ax
16978 0000110D 51                                   push    cx
16979 0000110E 52                                   push    dx
16980 0000110F E89501                               call    SetMediaForFormat
16981 00001112 3C01                                 cmp     al, 1
16982 00001114 7503                                 jnz     short WhileErr
16983 00001116 E83101                               call    SetDasd
16984                                   WhileErr:
16985 00001119 5A                                   pop     dx
16986 0000111A 59                                   pop     cx
16987 0000111B 58                                   pop     ax
```

```
16988 0000111C E2BD                     loop    FormatRetry
16989                                   FormatFailed:
16990 0000111E C606[AA05]01             mov     byte [had_format_error], 1
16991                                                   ; Set the format error flag
16992 00001123 80FC06                   cmp     ah, 6           ; DSK_CHANGELINE_ERR - convert change line
16993 00001126 7502                     jnz     short DoMapIt   ; Error to time out error
16994 00001128 B480                     mov     ah, 80h         ; DSK_TIMEOUT_ERR
16995                                   DoMapIt:
16996 0000112A E97CFC                   jmp     maperror
16997                                   ; ---------------------------------------------------------------------------
16998
16999                                   FormatOk:
17000 0000112D C606[AA05]00             mov     byte [had_format_error], 0 ; reset the format error flag
17001 00001132 C3                       retn
17002                                   ; ---------------------------------------------------------------------------
17003
17004                                   ; 16/10/2022
17005
17006                                   ; ============================================================================
17007                                   ;
17008                                   ; VerifyTrack:
17009                                   ;
17010                                   ; input: ES:di points to bds for drive
17011                                   ; ============================================================================
17012
17013                                             ; 24/12/2023 - Retro DOS 5.0
17014                                   VerifyTrack:
17015 00001133 1E                       push    ds
17016 00001134 C51E[1200]               lds     bx, [ptrsav]    ; DS:BX points to request header.
17017 00001138 C55F13                   lds     bx, [bx+19]     ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET]
17018
17019                                             ; Come here with DS:[BX] -> packet, ES:[DI] -> bds
17020
17021 0000113B 8B4F03                   mov     cx, [bx+3]      ; [bx+A_VERIFYPACKET.VP_CYLINDER]
17022 0000113E 8B4701                   mov     ax, [bx+1]      ; [bx+A_VERIFYPACKET.VP_HEAD]
17023 00001141 8B5705                   mov     dx, [bx+5]      ; [bx+A_FORMATPACKET.FP_TRACKCOUNT]
17024 00001144 8A1F                     mov     bl, [bx]        ; [bx+A_FORMATPACKET.FP_SPECIALFUNCTIONS]
17025                                                   ; Get option flag word
17026 00001146 1F                       pop     ds
17027 00001147 C606[2001]04             mov     byte [rflag], 4         ; romverify
17028 0000114C 890E[3301]               mov     [curtrk], cx
17029 00001150 A2[3201]                 mov     [curhd], al     ; ASSUME heads < 256
17030 00001153 8B0E[AA04]               mov     cx, [sectorspertrack]
17031
17032                                   ; Check specialfunctions to see if DO_FAST_FORMAT has been
17033                                   ; specified if not we should go to the normal track verification
17034                                   ; routine. If fast format has been specified we should get the
17035                                   ; number of tracks to be verified and check it to see if it is
17036                                   ; > 255. If it is then it is an error and we should go to
17037                                   ; VerifyTrack_Err. If not multiply the number of tracks by the
17038                                   ; sectors per track to get the total number of sectors to be
17039                                   ; verified. This should also be less than equal to 255
17040                                   ; otherwise we go to same error exit. If everything is okay
17041                                   ; we initialise cx to the total sectors. use ax as a temporary
17042                                   ; register.
17043
17044                                                   ; Special function requested?
17045 00001157 F6C302                   test    bl, 2           ; DO_FAST_FORMAT
17046 0000115A 7421                     jz      short NormVerifyTrack
17047 0000115C 89D0                     mov     ax, dx          ; Get ax = number of trks to verify
17048 0000115E 08E4                     or      ah, ah
17049 00001160 752C                     jnz     short VerifyTrack_Err ; #tracks > 255
17050 00001162 F6E1                     mul     cl
17051 00001164 08E4                     or      ah, ah
17052 00001166 7526                     jnz     short VerifyTrack_Err ; #sectors > 255
17053 00001168 89C1                     mov     cx, ax
17054                                   ; 24/12/2023
17055 0000116A 26F6453F01               test    byte [es:di+3Fh], 1 ; PCDOS 7.1 IBMBIO.COM
17056                                   ; 10/12/2022
17057                                   ;test   byte [es:di+35], 1
17058                                   ;;test  word [es:di+35], 1 ; [es:di+BDS.flags]
17059                                                   ; fnon_removable
17060 0000116F 740C                     jz      short NormVerifyTrack
17061                                                   ; Multitrack operation = on?
17062                                   ; 10/12/2022
17063                                   ; 19/10/2022
17064 00001171 F606[A004]80             test    byte [multrk_flag], 80h
17065                                   ;test   word [multrk_flag], 80h ; MULTI_TRK_ON
17066                                   ;;test  ds:multrk_flag,       80h ; MULTI_TRK_ON
17067 00001176 7405                     jz      short NormVerifyTrack
17068 00001178 C606[A704]01             mov     byte [multitrk_format_flag], 1
17069                                   NormVerifyTrack:
17070 0000117D 31C0                     xor     ax, ax          ; 1st sector
17071 0000117F 31DB                     xor     bx, bx
17072 00001181 891E[A804]               mov     [xfer_seg], bx ; Use 0:0 as the transfer address for verify
17073 00001185 E83F00                   call    TrackIo
17074 00001188 C606[A704]00             mov     byte [multitrk_format_flag], 0
17075 0000118D C3                       retn
17076                                   ; ---------------------------------------------------------------------------
17077
17078                                   VerifyTrack_Err:
17079 0000118E B401                     mov     ah, 1
17080 00001190 E916FC                   jmp     maperror
17081                                   ; ---------------------------------------------------------------------------
17082
17083                                   ; 16/10/2022
17084
17085                                   ; ============================================================================
17086                                   ;
17087                                   ; ReadTrack:
17088                                   ;
17089                                   ; input: ES:di points to bds for drive
17090                                   ;
17091                                   ; ============================================================================
17092
17093                                   ReadTrack:
17094 00001193 C606[2001]02             mov     byte [rflag], 2         ; romread
17095 00001198 EB05                     jmp     short ReadWriteTrack
17096                                   ; ---------------------------------------------------------------------------
17097
17098                                   WriteTrack:
17099
17100                                   ; ============================================================================
17101                                   ;
17102                                   ; WriteTrack:
17103                                   ;
17104                                   ; input: ES:di points to bds for drive
17105                                   ;
17106                                   ; ============================================================================
17107
17108 0000119A C606[2001]03             mov     byte [rflag], 3         ; romwrite
17109
17110                                             ; Fall into ReadWriteTrack
17111
```

```
                                ; ============================================================================
                                ;
                                ; readWriteTrack:
                                ;
                                ; input:
                                ;    ES:di points to bds for drive
                                ;    rFlag - 2 for read,3 for write
                                ;
                                ; ============================================================================

                                ReadWriteTrack:
                                        ; save bds pointer segment so we can use it to access
                                        ; our packet. Notice that this is not the standard register
                                        ; assignment for accessing packets

                                        ; 19/10/2022
0000119F 06                             push    es
000011A0 C41E[1200]                     les     bx, [ptrsav]   ; ES:BX -> to request header
000011A4 26C45F13                       les     bx, [es:bx+19] ; [es:bx+IOCTL_REQ.GENERICIOCTL_PACKET]
000011A8 268B4703                       mov     ax, [es:bx+3]  ; [es:bx+A_TRACKREADWRITEPACKET.TRWP_CYLINDER]
000011AC A3[3301]                       mov     [curtrk], ax
000011AF 268B4701                       mov     ax, [es:bx+1]  ; [es:bx+A_TRACKREADWRITEPACKET.TRWP_HEAD]
000011B3 A2[3201]                       mov     [curhd], al    ; Assume heads < 256!!!
000011B6 268B4705                       mov     ax, [es:bx+5]  ; [es:bx+A_TRACKREADWRITEPACKET.TRWP_FIRSTSECTOR]
000011BA 268B4F07                       mov     cx, [es:bx+7]  ; [es:bx+A_TRACKREADWRITEPACKET.TRWP_SECTORSTOREADWRITE]
000011BE 26C45F09                       les     bx, [es:bx+9]  ; [es:bx+A_TRACKREADWRITEPACKET.TRWP_TRANSFERADDRESS]
                                                                ; Get transfer address

                                        ; we just trashed our packet address, but we no longer care

000011C2 8C06[A804]                     mov     [xfer_seg], es ; Pass transfer       segment
000011C6 07                             pop     es

                                        ; Fall into TrackIo

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

                                ; 16/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)

                                ; ============================================================================
                                ;
                                ; TrackIo:
                                ;   performs track read/write/verify
                                ;
                                ;  input:
                                ;     rFlag       - 2 = read
                                ;                   3 = write
                                ;                   4 = verify
                                ;     AX   - Index into track table of first sector to io
                                ;     CX   - Number of sectors to io
                                ;     Xfer_Seg:BX - Transfer address
                                ;     ES:DI       - Pointer to bds
                                ;     CurTrk      - Current cylinder
                                ;     CurHd       - Current head
                                ;
                                ; ============================================================================

                                ; 16/03/2019 - Retro DOS v4.0

                                        ; 24/12/2023 - Retro DOS 5.0

                                        ; 19/10/2022
                                TrackIo:
                                                                ; Procedure `disk' will pop stack to
000011C7 8926[3501]                     mov     [spsav], sp    ; SpSav and return if error
000011CB E8A9F7                         call    checksingle    ; Ensure correct disk is in drv
000011CE 803E[A905]01                   cmp     byte [media_set_for_format], 1
                                                                ; See if we have already set  disk
000011D3 7407                           jz      short Dptalreadyset ; base table
000011D5 50                             push    ax             ; set up tables       and variables for i/o
000011D6 51                             push    cx
000011D7 E8A0F9                         call    iosetup
000011DA 59                             pop     cx
000011DB 58                             pop     ax
                                Dptalreadyset:                          ; Point si at the table entry of the
000011DC BE[AC04]                       mov     si, tracktable ; first sector to be io'd
                                        ; 24/12/2023
                                        ;add    ax, ax         ; PCDOS 7.1 IBMBIO.COM
                                        ;add    ax, ax
000011DF D1E0                           shl     ax, 1
000011E1 D1E0                           shl     ax, 1
000011E3 01C6                           add     si, ax

                                        ; WE WANT:
                                        ; CX to be the number of times we have to loop
                                        ; DX to be the number of sectors we read on each iteration

000011E5 BA0100                         mov     dx, 1

                                        ; 24/12/2023
000011E8 26F6453F08                     test    byte [es:di+3Fh], 8 ; PCDOS 7.1 IBMBIO.COM
                                        ; 12/12/2022
                                        ;test   byte [es:di+23h], 8
                                        ;;test  word [es:di+35], 8 ; [es:di+BDS.flags]
                                                                ; good_tracklayout
000011ED 7402                           jz      short ionextsector

000011EF 87CA                           xchg    dx, cx         ; HEY! We can read all secs in one blow
                                ionextsector:
000011F1 51                             push    cx
000011F2 52                             push    dx
000011F3 46                             inc     si
000011F4 46                             inc     si             ; Skip over the       cylinder and head in
                                                                ; the track table
000011F5 AC                             lodsb                  ; Get sector ID       from track table
000011F6 A2[3101]                       mov     [cursec], al

                                        ; assumptions for a fixed disk multi-track disk     i/o
                                        ; 1). In the input CX (# of sectors to go) to TrackIo,
                                        ;     only CL is valid.
                                        ; 2). Sector size should be set      to 512 bytes.
                                        ; 3). Good track layout

                                        ; 24/12/2023
000011F9 26F6453F01                     test    byte [es:di+3Fh], 1 ; PCDOS 7.1 IBMBIO.COM
                                        ; 12/12/2022
                                        ;test   byte [es:di+23h], 1
                                        ;;test  word [es:di+35], 1 ; [es:di+BDS.flags]
                                                                ; fnon_removable ; Fixed disk?
000011FE 7414                           jz      short IoRemovable ; No

                                        ; 12/12/2022
00001200 F606[A004]80                   test    byte [multrk_flag], 80h
                                        ;test   word [multrk_flag], 80h ; MULTI_TRK_ON
```

```
                                                                        ; Allow multi-track operation?
00001205 740D                           jz      short IoRemovable ; No,don't do that.
00001207 8916[2201]                     mov     [seccnt], dx
0000120B 89D0                           mov     ax, dx
0000120D E823FA                         call    Disk
00001210 5A                             pop     dx
00001211 59                             pop     cx
00001212 F8                             clc
00001213 C3                             retn
                                ; ---------------------------------------------------------------------------

                                IoRemovable:
00001214 AC                             lodsb                   ; Get sector size index       from track
                                                                ; table and save it
00001215 50                             push    ax
00001216 56                             push    si
00001217 1E                             push    ds              ; Save Bios_Data
00001218 50                             push    ax
00001219 8A26[2C01]                     mov     ah, [eot]       ; Preserve whatever might be in      ah
                                                                ; Fetch EOT while ds-> Bios_Data
0000121D C536[2D01]                     lds     si, [dpt]
00001221 884403                         mov     [si+3], al      ; [si+DISK_PARMS.DISK_SECTOR_SIZ]
00001224 886404                         mov     [si+4], ah      ; [si+DISK_PARMS.DISK_EOT]
00001227 58                             pop     ax
00001228 1F                             pop     ds
00001229 88D0                           mov     al, dl
0000122B A3[2201]                       mov     [seccnt], ax
0000122E E802FA                         call    Disk
00001231 5E                             pop     si              ; Advance buffer pointer by adding
                                                                ; sector size
                                        ;pop    ax
                                        ; 24/12/2023
00001232 59                             pop     cx

                                        ; 02/09/2023 (PCDOS 7.1)
                                        ;call   SectorSizeIndexToSectorSize
                                        ;mov    cl, al ; 24/12/2023
00001233 B88000                         mov     ax, 128
00001236 D3E0                           shl     ax, cl

00001238 01C3                           add     bx, ax
0000123A 5A                             pop     dx
0000123B 59                             pop     cx
0000123C E2B3                           loop    ionextsector
0000123E 803E[A905]01                   cmp     byte [media_set_for_format], 1
                                        ;jz     short NoNeedDone
                                        ; 12/12/2022
00001243 7404                           je      short NoNeedDone2
00001245 E877F9                         call    done            ; set time of last access, and reset
                                                                ; entries in Dpt.
                                NoNeedDone:
00001248 F8                             clc     ; not necessary ('done' clears cf) ; 24/12/2023
                                NoNeedDone2:
00001249 C3                             retn

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

                                ; ---------------------------------------------------------------------------
                                ;
                                ; The sector size in bytes needs to be converted to an index value for the ibm
                                ; rom. (0=>128,1=>256,2=>512,3=>1024). It is assumed that only these values
                                ; are permissible.
                                ;
                                ; On Input   AX contains sector size in bytes
                                ; On Output  AL Contains index
                                ; All other registers preserved
                                ;
                                ; ---------------------------------------------------------------------------

                                ; 02/09/2023
                                ;SectSizeToSectIndex:
                                ;           cmp     ah, 2           ; (0=>128,1=>256,2=>512,3=>1024)
                                ;                                   ; examine upper       byte only
                                ;           ja      short OneK
                                ;           mov     al, ah          ; value in AH is the index!
                                ;           retn

                                ; ---------------------------------------------------------------------------
                                ;
                                ;OneK:
                                ;           mov     al, 3
                                ;           retn

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

                                ; 02/09/2023
                                ;SectorSizeIndexToSectorSize:
                                ;           mov     cl, al
                                ;           mov     ax, 128
                                ;           shl     ax, cl
                                ;           retn

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

                                ; 16/10/2022

                                ; ---------------------------------------------------------------------------
                                ;
                                ; SetDASD
                                ;
                                ; Set up the rom for formatting.
                                ; we have to tell the rom bios what type of disk is in the drive.
                                ;
                                ; On Input   - ES:di - Points to bds
                                ;
                                ; ---------------------------------------------------------------------------

                                        ; 24/12/2023 - Retro DOS 5.0
                                        ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:129Bh)

                                        ; 19/10/2022
                                SetDasd:
0000124A 803E[AA05]01                   cmp     byte [had_format_error], 1 ;
                                                                ; See if we've previously set dasd type
0000124F 740C                           jz      short DoSetDasd
                                        ; 24/12/2023
00001251 26F6453F80                     test    byte [es:di+3Fh], 80h
                                        ; 10/12/2022
                                        ;test   byte [es:di+23h], 80h
                                        ;;test  word [es:di+23h], 80h ; [es:di+BDS.flags]
                                                                ; set_dasd_true
00001256 7446                           jz      short DasdHasBeenSet
                                        ; 24/12/2023
00001258 2680653F7F                     and     byte [es:di+3Fh], 7Fh
```

```
                                        ; 10/12/2022
                                        ;and    byte [es:di+23h], 7Fh
                                        ;;and   word [es:di+23h], 0FF7Fh ; [es:di+BDS.flags]
                                                                ; ~set_dasd_true
                                DoSetDasd:
0000125D C606[AA05]00                   mov     byte [had_format_error], 0 ; Reset it
00001262 C606[3B01]50                   mov     byte [gap_patch], 50h ; Format gap for 48tpi disks
00001267 B004                           mov     al, 4
                                        ; 24/12/2023
00001269 268A653E                       mov     ah, [es:di+3Eh]
                                        ; 02/09/2023
                                        ;mov    ah, [es:di+22h] ; [es:di+BDS.formfactor]
0000126D 80FC02                         cmp     ah, 2
                                        ;cmp    byte [es:di+22h], 2 ; [es:di+BDS.formfactor]
                                                                ; DEV_3INCH720KB
00001270 7414                           jz      short DoSet
                                        ; 24/12/2023
00001272 B001                           mov     al, 1
                                        ;cmp    ah, 1
00001274 38C4                           cmp     ah, al  ; 1
                                        ;cmp    byte [es:di+22h], 1 ; [es:di+BDS.formfactor]
                                                                ; DEV_5INCH96TPI
                                        ;jz     short GotBig
                                        ; 24/12/2023
                                        ;mov    al, 1
                                        ;jmp    short DoSet
                                        ; 02/09/2023
00001276 750E                           jnz     short DoSet
                                GotBig:
                                        ;mov    al, 2           ; 160/320k in a       1.2 meg drive
                                        ; 02/09/2023
00001278 40                             inc     ax  ; mov al, 2
00001279 803E[A805]00                   cmp     byte [mediatype], 0
0000127E 7506                           jnz     short DoSet
                                        ;mov    al, 3           ; 1.2meg in a 1.2meg drive
                                        ; 10/12/2022
                                        ;inc    al  ; al = 3
                                        ; 18/12/2022
00001280 40                             inc     ax  ; al = 3
00001281 C606[3B01]54                   mov     byte [gap_patch], 54h
                                DoSet:
00001286 1E                             push    ds
00001287 56                             push    si

                                        ;mov    ds, [zeroseg]  ; Point to interrupt vectors
                                        ; 02/09/2023
00001288 31F6                           xor     si, si
0000128A 8EDE                           mov     ds, si  ; 0

0000128C C5367800                       lds     si, [DSKADR]
                                        ;lds    si, [78h]       ; [DSKADR]  (Int 1Eh)
                                        ;;lds   si, ds:78h

00001290 C644090F                       mov     byte [si+9], 0Fh ;
                                                                ; [si+DISK_PARMS.DISK_HEAD_STTL]
00001294 5E                             pop     si
00001295 1F                             pop     ds
00001296 B417                           mov     ah, 17h
00001298 268A5504                       mov     dl, [es:di+4]
0000129C CD13                           int     13h             ; DISK - DISK -       SET TYPE (AT,XT2,XT286,CONV,PS
                                                                ; AL = disk type AL = 03h - high-capacity disk in high-capacity   drive
                                DasdHasBeenSet:
0000129E 268A6513                       mov     ah, [es:di+13h]       ; [es:di+BDS.secpertrack]
000012A2 8826[3701]                     mov     [formt_eot], ah
000012A6 C3                             retn

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

                                ; 16/10/2022

                                ; ---------------------------------------------------------------------------
                                ;
                                ; Set Media Type for Format
                                ; Performs the int 13 with ah = 18h to see if the medium described in the
                                ; BPB area in the BDS can be handled by the rom.
                                ; On Input, ES:DI -> current BDS.
                                ; The status of the operation is returned in AL
                                ;
                                ;   - 0 - if the support is available,and the combination is valid.
                                ;   - 1 - no rom support
                                ;   - 2 - illegal combination
                                ;   - 3 - no media present (rom support exists but cannot determine now)
                                ;
                                ; Flags also may be altered. All other registers preserved.
                                ; If the call to rom returns no error,then the current Dpt is "replaced" by
                                ; the one returned by the rom. This is Done by changing the pointer in [Dpt]
                                ; to the one returned. the original pointer to the disk base table is stored
                                ; in TempDpt, until it is restored.
                                ;
                                ; ---------------------------------------------------------------------------

                                        ; 24/12/2023 - Retro DOS 5.0

                                        ; 19/10/2022
                                SetMediaForFormat:
000012A7 51                             push    cx
000012A8 52                             push    dx

                                        ; If we have a format error, then do not change Dpt, TempDpt.
                                        ; but we need to call int 13h, ah=18h again.

000012A9 803E[AA05]01                   cmp     byte [had_format_error], 1
000012AE 7425                           jz      short SkipSaveDskAdr
000012B0 30C0                           xor     al, al          ; If already done return 0
000012B2 803E[A905]01                   cmp     byte [media_set_for_format], 1
000012B7 7502                           jnz     short DoSetMediaForFormat
000012B9 EB7D                           jmp     SetMediaRet     ; Media already set
                                ; ---------------------------------------------------------------------------

                                DoSetMediaForFormat:
000012BB 06                             push    es
000012BC 56                             push    si

                                        ; 02/09/2023
                                        ;mov    es, [zeroseg]  ; Point to interrupt vectors
000012BD 31F6                           xor     si, si ; 0
000012BF 8EC6                           mov     es, si

000012C1 26C4367800                     les     si, [es:DSKADR]
                                        ;les    si, es:78h      ; [es:DSKADR]
                                                                ; Get pointer to disk base table
000012C6 8936[2D01]                     mov     [dpt], si
000012CA 8C06[2F01]                     mov     [dpt+2], es     ; Save pointer to table

```

```
                                        ; 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
000012FD 31F6                           xor     si, si
000012FF 8EC6                           mov     es, si ; 0
00001301 06                             push    es ; * ; 02/09/2023

00001302 26C4367800                     les     si, [es:DSKADR]
                                        ;les    si, es:78h      ; [es:DSKADR] (Int 1Eh)
                                                                ; Get current disk base       table
00001307 8936[AB05]                     mov     [tempdpt], si
0000130B 8C06[AD05]                     mov     [tempdpt+2], es ; Save it

                                        ; 02/09/2023
                                        ;;mov   es, [zeroseg]
                                        ;xor    si, si ; 0
                                        ;mov    es, si
0000130F 07                             pop     es ; * ; 02/09/2023

                                        ;mov    es:78h, di
00001310 26893E7800                     mov     [es:DSKADR], di
                                        ;pop    word ptr es:7Ah         ; replace with one returned by rom
00001315 268F067A00                     pop     word [es:DSKADR+2]
0000131A C606[A905]01                   mov     byte [media_set_for_format], 1
                                skip_disk_base_setting:
0000131F 30C0                           xor     al, al          ; Legal combination + rom support code
                                        ;mov    ds:had_format_error, al       ; Reset the flag
00001321 A2[AA05]                       mov     [had_format_error], al
00001324 EB0E                           jmp     short PopStatRet
                                ; ---------------------------------------------------------------------------

                                FormaStatErr:
                                        ; 10/12/2022
00001326 B003                           mov     al, 3

00001328 80FC0C                         cmp     ah, 0Ch         ; DSK_ILLEGAL_COMBINATION
                                                                ; Illegal combination =       0Ch
0000132B 7406                           jz      short FormatStatIllegalComb
0000132D 80FC80                         cmp     ah, 80h         ; DSK_TIMEOUT_ERR
00001330 7402                           jz      short FormatStatTimeOut
                                        ; 10/12/2022
                                        ;dec    al
                                        ; 18/12/2022
00001332 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
00001333 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:
00001334 5F                             pop     di
00001335 5E                             pop     si
00001336 1F                             pop     ds
00001337 07                             pop     es
                                SetMediaRet:
00001338 5A                             pop     dx
00001339 59                             pop     cx
0000133A 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:
0000133B 50                             push    ax
```

```
                                        ; 02/09/2023
0000133C B80100                         mov     ax, 1 ; PCDOS 7.1
0000133F 3806[A905]                     cmp     [media_set_for_format], al ; 1
                                        ;cmp    byte [media_set_for_format], 1
                                                        ; Reset while formatting?
00001343 7503                           jnz     short ResetDisk_cont
                                                        ; Then verify operation in "fmt & vrfy"
                                        ;mov    byte [had_format_error], 1 ; Might have failed.
00001345 A2[AA05]                       mov     [had_format_error], al ; 1
                                ResetDisk_cont:
                                        ; 02/09/2023 (ah=0)
                                        ;xor    ah, ah          ; So signals that we had a format error
00001348 CD13                           int     13h             ; DISK - RESET DISK SYSTEM
                                                                ; DL = drive (if bit 7 is set both hard     disks and floppy disks reset)
0000134A C606[7600]FF                   mov     byte [step_drv], 0FFh ; -1
                                                                ; Zap up the speed
0000134F 58                             pop     ax
00001350 C3                             retn
                                ; =============== S U B   R O U T I N E =======================================
                                ; 16/10/2022
                                ; ---------------------------------------------------------------------------
                                ;
                                ; This routine sets up the drive parameter table with the values needed for
                                ; format,does an int 13. values in Dpt are restored after a verify is done.
                                ;
                                ; on entry  -      ES:DI - points to bds for the drive
                                ;                 Xfer_Seg:BX - points to trkbuf
                                ;                 AL    - number of sectors
                                ;                 AH    - int 13 function code
                                ;                 CL    - sector number for verify
                                ;                 DS    - Bios_Data
                                ;
                                ; ON EXIT   -      DS,DI,ES,BX remain unchanged.
                                ;                 AX and flags are the results of the int 13
                                ;
                                ; ---------------------------------------------------------------------------
                                        ; 24/12/2023 - Retro DOS 5.0
                                        ; 19/10/2022
                                ToRom:
00001351 53                             push    bx
00001352 56                             push    si
                                        ; Compaq bug fix - check whether we are using new ROM
                                        ; functionality to set up format, not merely if it exists.
                                        ; This was formerly a check against [new_rom]
00001353 F606[A905]01                   test    byte [media_set_for_format], 1
00001358 7534                           jnz     short GotValidDpt
0000135A 50                             push    ax
0000135B 06                             push    es              ; Save bds segment
                                        ; 24/12/2023
0000135C 26807D3E02                     cmp     byte [es:di+3Eh], 2
                                        ;cmp    byte [es:di+22h], 2 ; [es:di+BDS.formfactor]
                                                                ; ffSmall ; is it a 3.5" drive?
                                        ; 24/12/2023
                                        ;pushf ; not necessary       ; (Save the cmp result)
00001361 8E06[1A00]                     mov     es, [zeroseg]
                                        ;les    si, es:78h      ; Get pointer to disk base table
00001365 26C4367800                     les     si, [es:DSKADR]
                                        ;mov    word ptr ds:dpt, si
                                        ;mov    word ptr ds:dpt+2, es ;       Save pointer to table
0000136A 8936[2D01]                     mov     [dpt], si
0000136E 8C06[2F01]                     mov     [dpt+2], es     ; Save pointer to table

00001372 A0[3701]                       mov     al, [formt_eot]
00001375 26884404                       mov     [es:si+4], al  ; [es:si+DISK_PARMS.DISK_EOT]
00001379 A0[3B01]                       mov     al, [gap_patch]
0000137C 26884407                       mov     [es:si+7], al  ; [es:si+DISK_PARMS.DISK_FORMT_GAP]
                                                                ; Important for       format
00001380 26C644090F                     mov     byte [es:si+9], 0Fh ; [es:si+DISK_PARMS.DISK_HEAD_STTL]
                                                                ; Assume we are       doing a seek operation
                                                                ; Setup motor start correctly for 3.5" drives
                                        ; 24/12/2023
                                        ;popf                   ; Get result of       earlier cmp
00001385 7505                           jnz     short MotorStrtOK
00001387 26C6440A04                     mov     byte [es:si+0Ah], 4 ; [es:si+DISK_PARMS.DISK_MOTOR_STRT]
                                MotorStrtOK:
0000138C 07                             pop     es              ; Restore bds segment
0000138D 58                             pop     ax
                                GotValidDpt:
0000138E 8B16[3901]                     mov     dx, [trknum]    ; Set track number
00001392 88D5                           mov     ch, dl          ; Set low 8 bits in ch
00001394 268A5504                       mov     dl, [es:di+4]  ; Set drive number
00001398 8A36[3801]                     mov     dh, [hdnum]    ; Set head number
0000139C 06                             push    es              ; Save bds segment
0000139D 8E06[A804]                     mov     es, [xfer_seg]
000013A1 CD13                           int     13h             ; DISK -
000013A3 07                             pop     es              ; Restore bds segment
000013A4 5E                             pop     si
000013A5 5B                             pop     bx
000013A6 C3                             retn

                                ; ---------------------------------------------------------------------------
                                ; 16/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ; 24/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)

                                ; BIOSCODE:1124h (MSDOS 6.21, IO.SYS)
                                ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:1404h

                                ; ===========================================================================
                                ;
                                ; get the owner of the physical drive represented by the logical drive in al.
                                ; the assumption is that we **always** keep track of the owner of a drive!!
                                ; if this is not the case, the system may hang, just following the linked list.
                                ;
                                ; ===========================================================================

                                        ; 24/12/2023 - Retro DOS 5.0

                                        ; 19/10/2022
                                ioctl_getown:
000013A7 E8FAF1                         call    SetDrive
000013AA 268A4504                       mov     al, [es:di+4]  ; [es:di+BDS.drivenum]
                                                                ; Get physical drive number
000013AE C43E[1901]                     les     di, [start_bds] ; Get start of bds chain
                                ownloop:
000013B2 26384504                       cmp     [es:di+4], al  ; [es:di+BDS.drivenum]
000013B6 7507                           jnz     short getnextBDS
```

```
                                        ; 24/12/2023
000013B8 26F6453F20                     test    byte [es:di+3Fh], 20h ; (PCDOS 7.1 IBMBIO.COM)
                                        ; 10/12/2022
                                        ;test   byte [es:di+23h], 20h
                                        ;;test  word [es:di+23h], 20h ; [es:di+BDS.flags]
                                                                      ; fi_own_physical
000013BD 7514                           jnz     short exitown
                                getnextBDS:
000013BF 26C43D                         les     di, [es:di]     ; [es:di+BDS.link]
000013C2 EBEE                           jmp     short ownloop
                                ; ---------------------------------------------------------------------------

                                ; ===========================================================================
                                ;
                                ; set the ownership of the physical drive represented by the logical drive
                                ; in al to al.
                                ;
                                ; ===========================================================================

                                        ; 24/12/2023 - Retro DOS 5.0

                                        ; 19/10/2022
                                ioctl_setown:
000013C4 E8DDF1                         call    SetDrive
000013C7 C606[7A00]01                   mov     byte [fsetowner], 1
                                                                      ; set flag for CheckSingle to look at.
000013CC E8A8F5                         call    checksingle
                                        ; 02/09/2023
000013CF FE0E[7A00]                     dec     byte [fsetowner] ; 0
                                        ;mov    byte [fsetowner], 0
                                                                      ; set ownership       of drive reset flag
                                        ; Fall into ExitOwn

                                ; ===========================================================================
                                ;
                                ; if there is only one logical drive assigned to this physical drive, return
                                ; 0 to user to indicate this. Enter with ES:di -> the owner's bds.
                                ;
                                ; ===========================================================================

                                        ; 24/12/2023 - Retro DOS 5.0
                                exitown:
000013D3 30C9                           xor     cl, cl
                                        ; 24/12/2023
000013D5 26F6453F10                     test    byte [es:di+3Fh], 10h ; (PCDOS 7.1 IBMBIO.COM)
                                        ; 12/12/2022
                                        ;test   byte [es:di+23h], 10h
                                        ;;test  word [es:di+23h], 10h ; [es:di+BDS.flags]
                                                                      ; fi_am_mult
000013DA 7406                           jz      short exitnomult
000013DC 268A4D05                       mov     cl, [es:di+5]  ; [es:di+BDS.drivelet]
                                                                      ; Get logical drive number
                                                                      ; Get it 1-based
000013E0 FEC1                           inc     cl
                                exitnomult:
000013E2 C51E[1200]                     lds     bx, [ptrsav]
000013E6 884F01                         mov     [bx+1],cl       ; [bx+unit]
                                                                      ; Exit normal termination
                                        ; 12/12/2022
                                        ; cf=0
                                        ;clc
000013E9 C3                             retn

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

                                ; 16/10/2022

                                ; ---------------------------------------------------------------------------
                                ;
                                ; moves the old Dpt that had been saved in TempDpt back to Dpt. this is done
                                ; only if the first byte of TempDpt is not -1.
                                ; all registers (including flags) are preserved.
                                ;
                                ; ---------------------------------------------------------------------------

                                        ; 24/12/2023
                                        ; 19/10/2022
                                RestoreOldDpt:
                                        ; if we have already restored the disk base table earlier,
                                        ; do not do it again.

000013EA 50                             push    ax
000013EB 30C0                           xor     al, al
000013ED A2[AA05]                       mov     [had_format_error], al; Reset flag and
000013F0 8606[A905]                     xchg    al, [media_set_for_format] ; get current flag setting
000013F4 08C0                           or      al, al
000013F6 7418                           jz      short DontRestore
000013F8 56                             push    si
000013F9 1E                             push    ds
000013FA 06                             push    es
000013FB C536[AB05]                     lds     si, [tempdpt]

                                        ; 17/10/2022
                                        ;mov    es, [cs:BIOSDATAWORD]
                                        ;;mov   es, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
                                        ;mov    es, [es:zeroseg]
                                        ;;mov   es, es:zeroseg ; CAS -- bleeeech!

                                        ; 24/12/2023
000013FF 31C0                           xor     ax, ax
00001401 8EC0                           mov     es, ax ; 0

                                        ;mov    es:78h,si       ; [es:DSKADR]  (Int 1Eh)
00001403 2689367800                     mov     [es:DSKADR], si
                                        ;mov    word ptr es:7Ah, ds ; [es:DSKADR+2]
00001408 268C1E7A00                     mov     [es:DSKADR+2], ds
0000140D 07                             pop     es
0000140E 1F                             pop     ds
0000140F 5E                             pop     si
                                DontRestore:
00001410 58                             pop     ax
                                        ; 12/12/2022
                                        ; cf=0
                                        ;clc                    ;  Clear carry
00001411 C3                             retn

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

                                ; 16/10/2022

                                ; ===========================================================================
                                ;   get media id
                                ; ===========================================================================
                                ;
```

```
                        ; FUNCTION: get the volume label,the system id and the serial number from
                        ;       the media that has the extended boot record.
                        ;       for the conventional media,this routine will return "unknown
                        ;       media type" error to dos.
                        ;
                        ; INPUT :   ES:di -> bds table for this drive.
                        ;
                        ; OUTPUT:   the request packet filled with the information,if not carry.
                        ;       if carry set,then al contains the device driver error number
                        ;       that will be returned to dos.
                        ;       register DS,DX,AX,CX,DI,SI destroyed.
                        ;
                        ; SUBROUTINES TO BE CALLED:
                        ;   BootIo:NEAR
                        ;
                        ; LOGIC:
                        ;   to recognize the extended boot record,this logic will actually
                        ;   access the boot sector even if it is a hard disk.
                        ;   note:the valid extended bpb is recognized by looking at the mediabyte
                        ;   field of bpb and the extended boot signature.
                        ;
                        ; {
                        ;   get logical drive number from bds table;
                        ;   rFlag = read operation;
                        ;   BootIo;         /*get the media boot record into the buffer
                        ;   if (no error) then
                        ;        if (extended boot record) then
                        ;          { set volume label,volume serial number and system id
                        ;            of the request packet to those of the boot record;
                        ;          };
                        ;        else               /*not an extended bpb */
                        ;          { set register al to "unknown media.." error code;
                        ;            set carry bit;
                        ;          };
                        ;   else
                        ;        ret;      /*already error code is set in the register al
                        ;
                        ; ============================================================================

                        ;size_of_EXT_BOOT_SERIAL equ 4
                        ;;size_of_EXT_BOOT_VOL_LABEL equ 11
                        ;;size_of_EXT_SYSTEM_ID equ 8

                                ; 24/12/2023 - Retro DOS 5.0
                                ; (PCDOS 7.1 IBMBIO.COM - BIOSCODE:1478h)

                                ; 19/10/2022
                        GetMediaId:
00001412 E8B000                 call    ChangeLineChk
00001415 268A4505               mov     al, [es:di+5]  ; [es:di+BDS.drivelet] ; Logical drive number
00001419 C606[2001]02           mov     byte [rflag], 2        ; Read operation
0000141E E88C00                 call    BootIo         ; Read boot sector into       DiskSector
00001421 722E                   jb      short IOCtl_If1
                                                       ; Valid? (0F0h-0FFh?)
00001423 803E[6701]F0           cmp     byte [disksector+15h], 0F0h
                                                       ; [disksector+EXT_BOOT.BPB+EBPB.MEDIADESCRIPTOR]
                                ;jb     short IOCtl_If2        ; brif not valid (0F0h - 0FFh)
                                ; 24/12/2023
00001428 7225                   jb      short IOCtl_If7

                                ; 24/12/2023
                                ; 10/12/2022
                                ;mov    si, disksector+26h
                                ;;;
                                ; 24/12/2023
                                ;mov    si, disksector+43h ; BS_FAT32_VolID
0000142A BE[9401]               mov     si, disksector+42h ; BS_FAT32_BootSig ; 24/12/2023
0000142D 833E[6801]00           cmp     word [disksector+16h], 0 ; BPB.FATSz16
00001432 7403                   jz      short IOCtl_If3 ; FAT32 fs
00001434 83EE1C                 sub     si, 1Ch         ; FAT (12-16) fs ; 43h-1Ch = 27h ; BS_VolID
                                ; si = disksector+26h = BS_BootSig ; 24/12/2023
                        IOCtl_If3:
                                ;cmp    byte [si-1], 29h ; BS_BootSig or BS_FAT32_BootSig
                                ;;;
00001437 803C29                 cmp     byte [si], 29h
                                ;cmp    byte [disksector+26h], 29h ; [disksector+EXT_BOOT.SIG]
                                                       ; EXT_BOOT_SIGNATURE
0000143A 7512                   jne     short IOCtl_If2        ; not extended boot record
0000143C C43E[1200]             les     di, [ptrsav]   ; es:di points to request header
00001440 26C47F13               les     di, [es:bx+19] ; [es:bx+IOCTL_REQ.GENERICIOCTL_PACKET]
                                ; 10/12/2022
                                ;mov    si, disksector+27h ; disksector+EXT_BOOT.SERIAL
00001444 46                     inc     si
                                ; 24/12/2023
                                ; si = disksector+27h (BS_VolID)
                                ;      or disksector+43h (BS_FAT32_VolID)

00001445 83C702                 add     di, 2          ; A_MEDIA_ID_INFO.MI_SERIAL
                        IOCtl_If4:      ; 24/12/2023
00001448 B91700                 mov     cx, 23         ; size_of_EXT_BOOT_SERIAL
                                                       ; L+size_of_EXT_BOOT_VOL_LABEL
                                                       ; +size_of_EXT_SYSTEM_ID
0000144B F3A4                   rep movsb              ; Move from Bios_Data into request packet

                                ; 10/12/2022
                                ; cf = 0
                                ;clc

0000144D C3                     retn
                        ; ---------------------------------------------------------------------------

                                ; 24/12/2023
                        IOCtl_If2:
0000144E F9                     stc
                        IOCtl_If7:
0000144F B007                   mov     al, 7          ; error_unknown_media
                                ;stc
                        IOCtl_If6:
                        IOCtl_If1:
00001451 C3                     retn
                        ; ---------------------------------------------------------------------------

                        ; 16/10/2022

                        ; ============================================================================
                        ;  set media id
                        ; ============================================================================

                        ; function: set the volume label, the system id and the serial number of
                        ;       the media that has the extended boot record.
                        ;       for the conventional media, this routine will return "unknown
                        ;       media.." error to dos.
                        ;       this routine will also set the corresponding informations in
                        ;       the bds table.
```

```
17979                                   ;
17980                                   ; input :   ES:di -> bds table for this drive.
17981                                   ;
17982                                   ; output:   the extended boot record in the media will be set according to
17983                                   ;       the request packet.
17984                                   ;       if carry set, then al contains the device driver error number
17985                                   ;       that will be returned to dos.
17986                                   ;
17987                                   ; subroutines to be called:
17988                                   ;   BootIo:NEAR
17989                                   ;
17990                                   ; logic:
17991                                   ;
17992                                   ; {
17993                                   ;   get drive_number from bds;
17994                                   ;   rFlag = "read operation";
17995                                   ;   BootIo;
17996                                   ;   if (no error) then
17997                                   ;        if (extended boot record) then
17998                                   ;          { set volume label,volume serial number and system id
17999                                   ;            of the boot record to those of the request packet;
18000                                   ;            rFlag = "write operation";
18001                                   ;            get drive number from bds;
18002                                   ;            BootIo;        /*write it back*/
18003                                   ;          };
18004                                   ;        else               /*not an extended bpb */
18005                                   ;          { set register al to "unknown media.." error code;
18006                                   ;            set carry bit;
18007                                   ;            ret;  /*return back to caller */
18008                                   ;          };
18009                                   ;   else
18010                                   ;        ret;              /*already error code is set */
18011                                   ;
18012                                   ; ==========================================================================
18013
18014                                             ; 24/12/2023 - Retro DOS 5.0
18015
18016                                             ; 19/10/2022
18017                                   SetMediaId:
18018 00001452 E87000                             call    ChangeLineChk
18019 00001455 268A4505                           mov     al, [es:di+5]  ; [es:di+BDS.drivelet]
18020                                                                    ; Logical drive       number
18021 00001459 88C2                               mov     dl, al
18022 0000145B C606[2001]02                       mov     byte [rflag], 2       ; romread
18023 00001460 52                                 push    dx
18024 00001461 E84900                             call    BootIo         ; Read boot sec       to Bios_Data:DiskSector
18025 00001464 5A                                 pop     dx
18026 00001465 72EA                               jb      short IOCtl_If6
18027                                                                    ; Valid? (0F0h-0FFh?)
18028 00001467 803E[6701]F0                       cmp     byte [disksector+15h], 0F0h
18029                                                                    ; [disksector+EXT_BOOT.BPB+EBPB.MEDIADESCRIPTOR]
18030 0000146C 72E1                               jb      short IOCtl_If7       ; Brif not
18031
18032                                             ; 24/12/2023
18033                                             ;cmp    byte [disksector+26h], 29h ; [disksector+EXT_BOOT.SIG]
18034                                             ;                       ; EXT_BOOT_SIGNATURE
18035                                             ;jnz    short IOCtl_If7       ; not extended boot record
18036
18037 0000146E 06                                 push    es             ; Save BDS pointer
18038 0000146F 57                                 push    di
18039 00001470 1E                                 push    ds             ; Point ES To boot record
18040 00001471 07                                 pop     es
18041
18042                                             ; 24/12/2023
18043                                             ;;;
18044                                             ;mov    di, disksector+43h ; disksector+EXT_BOOT.SERIAL
18045 00001472 BF[9401]                           mov     di, disksector+42h ; BS_FAT32_BootSig ; 24/12/2023
18046 00001475 833E[6801]00                       cmp     word [disksector+16h], 0 ; BPB.FATSz16
18047 0000147A 7403                               jz       short IOCtl_If5       ; FAT32 fs
18048 0000147C 83EF1C                             sub     di, 1Ch        ; 67-28 ; offset disksektor+27h
18049                                             ; di = disksector+26h = BS_BootSig ; 24/12/2023
18050                                   IOCtl_If5:
18051                                             ;cmp    byte [di-1], 29h ; BS_BootSig or BS_FAT32_BootSig
18052 0000147F 803D29                             cmp     byte [di], 29h
18053 00001482 7404                               je      short IOCtl_If8
18054 00001484 5F                                 pop     di             ; not extended boot record
18055 00001485 07                                 pop     es
18056                                             ;jmp    short IOCtl_If7
18057                                             ; 24/12/2023
18058 00001486 EBC6                               jmp     short IOCtl_If2
18059                                   IOCtl_If8:
18060                                             ;;;
18061                                             ; 24/12/2023
18062                                             ;mov    di, disksector+27h ; disksector+EXT_BOOT.SERIAL
18063 00001488 47                                 inc     di
18064                                             ; di = disksector+27h (BS_VolID)
18065                                             ;       or disksector+43h (BS_FAT32_VolID)
18066
18067 00001489 C536[1200]                         lds     si, [ptrsav]   ; ds:si points to request header.
18068 0000148D C57413                             lds     si, [si+19]    ; [si+IOCTL_REQ.GENERICIOCTL_PACKET]
18069 00001490 83C602                             add     si, 2          ; A_MEDIA_ID_INFO.MI_SERIAL
18070
18071                                             ; 24/12/2023
18072                                             ;mov    cx, 23         ; size_of_EXT_BOOT_SERIAL
18073                                             ;                      ; +size_of_EXT_BOOT_VOL_LABEL
18074                                             ;                      ; +size_of_EXT_SYSTEM_ID
18075                                             ;rep movsb
18076 00001493 E8B2FF                             call    IOCtl_If4       ; copy volume serial, label and system id
18077
18078 00001496 06                                 push    es             ; point ds back to Bios_Data
18079 00001497 1F                                 pop     ds
18080 00001498 5F                                 pop     di             ; restore bds pointer
18081 00001499 07                                 pop     es
18082 0000149A E8B2F3                             call    mov_media_ids  ; update the bds media id info.
18083 0000149D 88D0                               mov     al, dl
18084 0000149F C606[2001]03                       mov     byte [rflag], 3       ; romwrite
18085 000014A4 E80600                             call    BootIo         ; write it back.
18086 000014A7 C606[1E01]FF                       mov     byte [tim_drv], 0FFh
18087                                                                    ; make sure chk_media check the driver
18088                                                                    ; return with error code from BootIo
18089 000014AC C3                                 retn
18090                                   ; ---------------------------------------------------------------------------
18091
18092                                             ; 24/12/2023
18093                                   ;IOCtl_If7:
18094                                   ;         mov     al, 7          ; error_unknown_media
18095                                   ;         stc
18096                                   ;IOCtl_If6:
18097                                   ;         retn
18098
18099                                   ; =============== S U B   R O U TI N E =======================================
18100
18101                                   ; 16/10/2022
18102
```

```
18103                                  ; ---------------------------------------------------------------------------
18104                                  ;   BootIo
18105                                  ; ---------------------------------------------------------------------------
18106                                  ;
18107                                  ; function: read/write the boot record into boot sector.
18108                                  ;
18109                                  ; input :
18110                                  ;       al=logical drive number
18111                                  ;       rFlag = operation (read/write)
18112                                  ;
18113                                  ; output:   for read operation,the boot record of the drive specified in bds
18114                                  ;       be read into the DiskSector buffer.
18115                                  ;       for write operation,the DiskSector buffer image will be written
18116                                  ;       to the drive specified in bds.
18117                                  ;       if carry set,then al contains the device driver error number
18118                                  ;       that will be returned to dos.
18119                                  ;       AX,CX,DX register destroyed.
18120                                  ;       if carry set,then al will contain the error code from DiskIO.
18121                                  ;
18122                                  ; subroutines to be called:
18123                                  ;   DiskIO:NEAR
18124                                  ;
18125                                  ; logic:
18126                                  ;
18127                                  ; {
18128                                  ;   first_sector = 0;       /*logical sector 0 is the boot sector */
18129                                  ;   sectorcount = 1;        /*read 1 sector only */
18130                                  ;   buffer = DiskSector;    /*read it into the DiskSector buffer */
18131                                  ;   call DiskIO (rFlag,drive_number,first_sector,sectorcount,buffer);
18132                                  ; }
18133                                  ; ===========================================================================
18134
18135                                              ; 19/10/2022
18136                                  BootIo:
18137 000014AD 06                               push    es
18138 000014AE 57                               push    di
18139 000014AF 53                               push    bx
18140 000014B0 1E                               push    ds
18141 000014B1 07                               pop     es              ; Point ES: to Bios_Data
18142
18143                                           ; Call DiskIO to read/write the boot sec. The parameters which
18144                                           ; need to be initialized for this subroutine out here are
18145                                           ; - Transfer address to Bios_Data:DiskSector
18146                                           ; - Low sector needs to be initalized to 0. this is a reg. param
18147                                           ; - Hi sector in [Start_Sec_H] needs to be initialised to 0.
18148                                           ; - Number of sectors <-- 1
18149
18150 000014B2 BF[5201]                         mov     di, disksector ; es:di -> transfer address
18151 000014B5 31D2                             xor     dx, dx          ; First sector (h) -> 0
18152 000014B7 8916[9C04]                       mov     [start_sec_h], dx ; Start sector (h) -> 0
18153 000014BB B90100                           mov     cx, 1
18154 000014BE E850F5                           call    diskio
18155 000014C1 5B                               pop     bx
18156 000014C2 5F                               pop     di
18157 000014C3 07                               pop     es
18158 000014C4 C3                               retn
18159
18160                                  ; =============== S U B   R O U T I N E =======================================
18161
18162                                  ; 16/10/2022
18163
18164                                  ; ---------------------------------------------------------------------------
18165                                  ;   ChangeLineChk
18166                                  ; ---------------------------------------------------------------------------
18167                                  ;
18168                                  ; when the user calls get/set media id call before dos establishes the media
18169                                  ; by calling "media_chk",the change line activity of the drive is going to be
18170                                  ; lost.     this routine will check the change line activity and will save the
18171                                  ; history in the flags.
18172                                  ;
18173                                  ; FUNCTION: check the change line error activity
18174                                  ;
18175                                  ; INPUT :  ES:di -> bds table.
18176                                  ;
18177                                  ; OUTPUT:   flag in bds table will be updated if change line occurs.
18178                                  ;
18179                                  ; SUBROUTINES TO BE CALLED:
18180                                  ;   Set_Changed_DL
18181                                  ;
18182                                  ; ---------------------------------------------------------------------------
18183
18184                                              ; 24/12/2023 - Retro DOS 5.0
18185                                  ChangeLineChk:
18186 000014C5 268A5504                         mov     dl, [es:di+4]  ; [es:di+BDS.drivenum]
18187 000014C9 08D2                             or      dl, dl          ; Fixed disk?
18188 000014CB 7821                             js      short ChangeLnChkRet ; Yes, skip it.
18189                                           ; 24/12/2023
18190 000014CD 26F6453F04                       test    byte [es:di+3Fh], 4 ; [es:di+BDS.flags] ; PCDOS 7.1
18191                                           ; 12/12/2022
18192                                           ;test   byte [es:di+23h], 4
18193                                           ;;test  word [es:di+23h], 4 ; [es:di+BDS.flags]
18194                                                                   ; return_fake_bpb
18195 000014D2 751A                             jnz     short ChangeLnChkRet
18196 000014D4 803E[7700]01                     cmp     byte [fhave96], 1   ; This rom support change line?
18197 000014D9 7513                             jnz     short ChangeLnChkRet
18198 000014DB E8AD07                           call    haschange       ; This drive support change line?
18199 000014DE 740E                             jz      short ChangeLnChkRet ; Do nothing
18200
18201                                           ; Execute the rom disk interrupt to check changeline activity.
18202
18203 000014E0 B416                             mov     ah, 16h
18204 000014E2 CD13                             int     13h     ; DISK - FLOPPY       DISK - CHANGE OF DISK STATUS (AT,XT2,XT286,CONV,PS)
18205                                                           ; DL = drive to       check
18206                                                           ; Return: AH = disk change status
18207 000014E4 7308                             jnb     short ChangeLnChkRet
18208 000014E6 53                               push    bx
18209 000014E7 BB4000                           mov     bx, 40h         ; fchanged
18210                                                                   ; Update flag in BDS for this
18211                                                                   ; physical drive
18212 000014EA E87707                           call    set_changed_dl
18213 000014ED 5B                               pop     bx
18214                                  ChangeLnChkRet:
18215 000014EE C3                               retn
18216
18217                                  ; ---------------------------------------------------------------------------
18218
18219                                  ; 16/10/2022
18220
18221                                  ; ===========================================================================
18222                                  ;   GetAccessFlag
18223                                  ; ===========================================================================
18224                                  ;
18225                                  ; FUNCTION: get the status of UNFORMATTED_MEDIA bit of flags in bds table
18226                                  ;
```

```
                                ; INPUT :
                                ;       ES:di -> bds table
                                ;
                                ; OUTPUT:   a_DiskAccess_Control.dac_access_flag = 0 if disk i/o not allowed.
                                ;                                                 = 1 if disk i/o allowed.
                                ; ============================================================================

                                          ; 24/12/2023 - Retro DOS 5.0

                                          ; 19/10/2022
                                GetAccessFlag:
000014EF C51E[1200]                       lds     bx, [ptrsav]   ; DS:BX points to request header
000014F3 C55F13                           lds     bx, [bx+19]    ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET]
                                          ;mov    al, 0          ; Assume result       is unformatted
                                          ; 10/12/2022
000014F6 28C0                             sub     al, al
                                          ; 24/12/2023
000014F8 26F6454002                       test    byte [es:di+40h], 02h ; (PCDOS 7.1 IBMBIO.COM)
                                          ;test   word ptr es:[di+3Fh], 200h
                                          ; 10/12/2022
                                          ;test   byte [es:di+36], 02h
                                          ;;test  word [es:di+35], 200h ; [es:di+BDS.flags]
                                                                  ; unformatted_media
000014FD 7501                             jnz     short GafDone  ; Done if unformatted
                                          ;inc    al             ; Return true for formatted
                                          ; 24/12/2023
000014FF 40                               inc     ax
                                GafDone:
00001500 884701                           mov     [bx+1], al     ; [bx+A_DISKACCESS_CONTROL.DAC_ACCESS_FLAG]
00001503 C3                               retn
                                ; ---------------------------------------------------------------------------

                                ; 16/10/2022

                                ; ============================================================================
                                ;   SetAccessFlag
                                ; ============================================================================
                                ;
                                ; function: set/reset the UNFORMATTED_MEDIA bit of flags in bds table
                                ;
                                ; input :
                                ;       ES:di -> bds table
                                ;
                                ; output:   unformtted_media bit modified according to the user request
                                ; ============================================================================

                                          ; 24/12/2023 - Retro DOS 5.0

                                          ; 19/10/2022
                                SetAccessFlag:
00001504 C51E[1200]                       lds     bx, [ptrsav]   ; ES:BX points to request header
00001508 C55F13                           lds     bx, [bx+19]    ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET]
                                          ; 24/12/2023
0000150B 26806540FD                       and     byte [es:di+40h], 0FDh ; (PCDOS 7.1 IBMBIO.COM)
                                          ;and    word ptr es:[di+3Fh], 0FDFFh
                                          ; 10/12/2022
                                          ;and    byte [es:di+36], 0FDh
                                          ;;and   word [es:di+35], 0FDFFh ; [es:di+BDS.flags]
                                                                  ; ~unformatted_media
00001510 807F0100                         cmp     byte [bx+1], 0 ; [bx+A_DISKACCESS_CONTROL.DAC_ACCESS_FLAG]
00001514 7505                             jnz     short saf_Done
                                          ; 24/12/2023
00001516 26804D4002                       or      byte [es:di+40h], 02h ; (PCDOS 7.1 IBMBIO.COM)
                                          ;or     word ptr es:[di+3Fh], 200h
                                          ; 15/04/2024
                                          ; 10/12/2022
                                          ;or     byte [es:di+36], 02h
                                          ;;or    word [es:di+35], 200h ; [es:di+BDS.flags]
                                                                  ; unformatted_media
                                saf_Done:
0000151B C3                               retn
                                ; ---------------------------------------------------------------------------

                                ; 16/10/2022

                                ; ============================================================================
                                ; Ioctl_Support_Query
                                ; ============================================================================
                                ;
                                ; New device command which was added in DOS 5.00 to allow a query of a
                                ; specific GENERIC IOCtl to see if it is supported. Bit 7 in the
                                ; device attributes specifies if this function is supported.
                                ;
                                ; ============================================================================

                                          ; 24/12/2023 - Retro DOS 5.0

                                          ; 19/10/2022
                                ioctl_support_query:
0000151C 06                               push    es
0000151D C41E[1200]                       les     bx, [ptrsav]   ; ES:BX Points to request header.
00001521 268B470D                         mov     ax, [es:bx+13] ; [es:bx+IOCTL_REQ.MAJORFUNCTION]
                                                                  ; AL == Major, AH == Minor
                                          ; 24/12/2023
                                          ; 02/09/2023 (PCDOS 7.1)
00001525 3C48                             cmp     al, 48h        ; IOC_NEW_DC (PCDOS 7.1)
                                                                  ; new generic ioctl function (FAT32)
00001527 7404                             je      short ioctl_support

00001529 3C08                             cmp     al, 8          ; IOC_DC
                                                                  ; See if major code is 8
0000152B 7513                             jne     short nosupport
                                ioctl_support:
0000152D 0E                               push    cs
0000152E 07                               pop     es
                                          ; 24/12/2023
                                          ; 02/09/2023
0000152F B90E00                           mov     cx, 14          ; (PCDOS 7.1) IOC_DC_TABLE_LEN
                                          ;mov    cx, 11         ; IOC_DC_TABLE_LEN
                                          ; 10/12/2022
00001532 BF[BD0E]                         mov     di, IOC_DC_Table
                                          ;mov    di, 0C60h      ; IOC_DC_Table
                                                                  ; at 2C7h:0C60h       = 70h:31D0h
00001535 86E0                             xchg    al, ah         ; Put minor code in AL
00001537 F2AE                             repne scasb            ; Scan for minor code in AL
00001539 7505                             jnz     short nosupport       ; it was not found
0000153B B80001                           mov     ax, 100h
                                          ; 10/12/2022
                                          ; (jump to ioctlsupexit is not required)
                                          ;jmp    short $+2      ; ioctlsupexit
                                                                  ; Signal ioctl is supported
                                          ;;jmp   short ioctlsupexit
                                ; ---------------------------------------------------------------------------
                                ioctlsupexit:
```

```
18351 0000153E 07                               pop     es
18352                                           ; 10/12/2022
18353                                           ; cf = 0
18354                                           ;clc
18355 0000153F C3                               retn
18356                                   ; ---------------------------------------------------------------------------
18357                                   nosupport:
18358 00001540 07                               pop     es
18359 00001541 E991EB                           jmp     bc_cmderr
18360                                   ; ---------------------------------------------------------------------------
18361
18362                                   ; 16/10/2022
18363
18364                                   ; ============================================================================
18365                                   ;   GetMediaSenseStatus
18366                                   ; ============================================================================
18367                                   ;
18368                                   ; FUNCTION: Will return the type of diskette media in the specified DOS
18369                                   ;       diskette drive and whether the media is the default type
18370                                   ;       for that drive. (default type means the max size for that
18371                                   ;       drive)
18372                                   ;
18373                                   ; INPUT :   ES:DI -> BDS table
18374                                   ; OUTPUT:   If carry clear
18375                                   ;       DS:BX -> Updated IOCtlPacket
18376                                   ;
18377                                   ;                   Special Function at offset 0:
18378                                   ;                         0       - Media detected is not default type
18379                                   ;                         1       - Media detected is default type
18380                                   ;
18381                                   ;                   Device Type at offset 1:
18382                                   ;                         2        - 720K 3.5" 80 tracks
18383                                   ;                         7       - 1.44M 3.5" 80 tracks
18384                                   ;                         9       - 2.88M 3.5" 80 tracks
18385                                   ;
18386                                   ; Error Codes returned in AX if carry set:
18387                                   ;
18388                                   ; 8102 - Drive not ready  - No disk is in the drive.
18389                                   ; 8107 - Unknown media type       - Drive doesn't support this function or
18390                                   ;                           the media is really unkown, any error
18391                                   ;                           other than "media not present"
18392                                   ;
18393                                   ; ============================================================================
18394
18395                                           ; 19/10/2022
18396                                   SenseMediaType:
18397 00001544 C51E[1200]                       lds     bx, [ptrsav]    ; DS:BX points to request header.
18398 00001548 C55F13                           lds     bx, [bx+19]     ; bx+IOCTL_REQ.GENERICIOCTL_PACKET]
18399                                           ; 10/10/2022
18400                                           ;mov    word [bx], 0    ; Initialize the 2 packet bytes
18401 0000154B 31D2                             xor     dx, dx
18402 0000154D 8917                             mov     [bx], dx ; 0
18403                                           ;
18404 0000154F 268A5504                         mov     dl, [es:di+4]   ; [es:di+BDS.drivenum]
18405                                                                   ; Get int 13h drive number from      BDS
18406                                           ; 10/12/2022
18407                                           ;xor    dh, dh          ; DX = physical       drive number
18408 00001553 B420                             mov     ah, 20h         ; Get Media Type function
18409                                                                   ; If no carry media type in AL
18410 00001555 CD13                             int     13h             ; DISK - QCACHE       - DISMOUNT
18411 00001557 7216                             jc      short MediaSenseEr ; error code      in AH
18412 00001559 FE07                             inc     byte [bx]       ; Signal media type is default (bit 1)
18413                                   DetermineMediaType:
18414 0000155B FEC8                             dec     al
18415 0000155D 3C02                             cmp     al, 2           ; Chk for 720K ie: (3-1) = 2
18416 0000155F 740A                             jz      short GotMediaType
18417 00001561 0404                             add     al, 4
18418 00001563 3C07                             cmp     al, 7           ; Chk for 1.44M ie: (4-1+4) = 7
18419 00001565 7404                             jz      short GotMediaType
18420 00001567 3C09                             cmp     al, 9           ; Chk for 2.88M       ie: (6-1+4) = 9
18421 00001569 7510                             jnz     short UnknownMediaType ; Just didn't recognize media type
18422                                   GotMediaType:
18423 0000156B 884701                           mov     [bx+1],al       ; Save the return value
18424                                           ; 10/12/2022
18425                                           ; cf = 0
18426                                           ;clc                    ; Signal success
18427 0000156E C3                               retn
18428                                   ; ---------------------------------------------------------------------------
18429
18430                                   MediaSenseEr:
18431 0000156F 80FC32                           cmp     ah, 32h         ; See if not default media error
18432 00001572 74E7                             jz      short DetermineMediaType ; Not really an error
18433 00001574 B002                             mov     al, 2           ; Now assume drive not ready
18434 00001576 80FC31                           cmp     ah, 31h         ; See if media was present
18435 00001579 7402                             jz      short SenseErrExit ; Return drive not ready
18436                                   UnknownMediaType:
18437 0000157B B007                             mov     al, 7           ; Just don't know the media type
18438                                   SenseErrExit:
18439 0000157D B481                             mov     ah, 81h         ; Signal error return
18440 0000157F F9                               stc
18441 00001580 C3                               retn
18442
18443                                   ; ---------------------------------------------------------------------------
18444                                           ; 10/12/2022
18445                                           ;db    0
18446                                   ; ---------------------------------------------------------------------------
18447
18448                                   ;---------------------------------------------------------------------------
18449                                   ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:15F2h
18450                                   ;---------------------------------------------------------------------------
18451                                   ; 26/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1)
18452
18453                                   ; =============== S U B R O U T I N E =======================================
18454
18455                                   SetLockState:
18456 00001581 C51E[1200]                       lds     bx, [ptrsav]    ; set media lock state
18457 00001585 C55F13                           lds     bx, [bx+13h]    ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET]
18458                                           ;mov    dl, [es:di+4]   ; [es:di+BDS.drivenum]
18459                                           ;call   check_int13h_exts_present
18460                                           ; 26/12/2023
18461 00001588 E82100                           call    check_int13h_exts_p
18462                                           ;mov    al, 3           ; unknown command error
18463 0000158B 721C                             jc      short setlockst_ret
18464 0000158D 8A07                             mov     al, [bx]        ; [bx+A_LOCKSTATE_CONTROL.LOCKSTATE_FUNCTIONS]
18465 0000158F B445                             mov     ah, 45h
18466 00001591 CD13                             int     13h             ; DISK - IBM/MS Extension - LOCK/UNLOCK DRIVE
18467                                           ; (DL - drive, [SI - disk address packet)
18468 00001593 884701                           mov     [bx+1], al      ; 1 = locked, 0 = not locked
18469                                           ; [bx+A_LOCKSTATE_CONTROL.LOCKSTATE_FLAG]
18470
18471                                           ; 26/12/2023
18472 00001596 EB0A                             jmp     short sls_em
18473
18474                                   ;       jnc     short setlockst_ret
```

```
18475                                   ;           mov     al, ah
18476                                   ;           call    maperror
18477                                   ;setlockst_ret:
18478                                   ;           mov     ah, 81h         ; Return this status in case of carry
18479                                   ;           retn
18480
18481                                   ; =============== S U B R O U T I N E =======================================
18482
18483                                   EjectMedia:
18484                                               ;mov    dl, [es:di+4]  ; eject media in drive
18485                                               ;                      ; [es:di+BDS.drivenum]
18486                                               ;call   check_int13h_exts_present
18487                                               ; 26/12/2023
18488 00001598 E81100                             call    check_int13h_exts_p
18489                                               ;mov    al, 3          ; unknown command error
18490 0000159B 720C                               jc      short ejectm_ret
18491 0000159D B80046                             mov     ax, 4600h
18492 000015A0 CD13                               int     13h             ; DISK - IBM/MS Extension - EJECT MEDIA
18493                                               ; (DL - drive)
18494                                   sls_em:                     ; 26/12/2023
18495 000015A2 7305                               jnc     short ejectm_ret
18496 000015A4 88E0                               mov     al, ah
18497 000015A6 E800F8                             call    maperror
18498                                   setlockst_ret:              ; 26/12/2023
18499                                   ejectm_ret:
18500 000015A9 B481                               mov     ah, 81h         ; Return this status in case of carry
18501 000015AB C3                                 retn
18502
18503                                   ; =============== S U B R O U T I N E =======================================
18504
18505                                               ; 26/12/2023
18506                                   check_int13h_exts_p:
18507 000015AC 268A5504                           mov     dl, [es:di+4]
18508
18509                                   check_int13h_exts_present:
18510 000015B0 B441                               mov     ah, 41h
18511 000015B2 53                                 push    bx
18512 000015B3 BBAA55                             mov     bx, 55AAh
18513 000015B6 CD13                               int     13h             ; DISK - Check for INT 13h Extensions
18514                                                                     ; BX = 55AAh, DL = drive number
18515                                                                     ; Return: CF set if not supported
18516                                                                     ; AH = extensions version
18517                                                                     ; BX = AA55h
18518                                                                     ; CX = Interface support bit map
18519 000015B8 81FB55AA                           cmp     bx, 0AA55h
18520 000015BC 5B                                 pop     bx
18521 000015BD 7505                               jnz     short exts_notsupported
18522 000015BF F6C102                             test    cl, 2           ; bit 1 - drive locking and ejecting subset
18523 000015C2 7503                               jnz     short exts_supported
18524                                   exts_notsupported:
18525                                               ; 26/12/2023
18526 000015C4 B003                               mov     al, 3
18527                                               ;
18528 000015C6 F9                                 stc
18529                                   exts_supported:
18530 000015C7 C3                                 retn
18531
18532                                   ; =============== S U B R O U T I N E =======================================
18533
18534                                   GetDrvMapInfo:
18535 000015C8 8CD9                               mov     cx, ds          ; get drive map information
18536                                                                     ;
18537                                                                     ; es:di points to BDS which belongs to
18538                                                                     ;       the requested logical/dos drive number
18539                                                                     ;
18540                                                                     ; Format of parameter block:
18541                                                                     ; Offset  Description (Table 01570)
18542                                                                     ;  00h    (call) length of this buffer (in bytes)
18543                                                                     ;  01h    (ret) number of bytes in parameter block
18544                                                                     ;          actually used
18545                                                                     ;  02h    (ret) drive flags
18546                                                                     ;  03h    (ret) physical drive number
18547                                                                     ;          00h-7Fh floppy
18548                                                                     ;          80h-FEh hard
18549                                                                     ;          FFh no physical drive
18550                                                                     ;  04h    (ret) bitmap of logical drives associated with
18551                                                                     ;          physical drive
18552                                                                     ;          bit 0 = drive A:, etc.
18553                                                                     ;  08h    (ret) relative block address of partition start
18554                                                                     ;          qword
18555                                                                     ;
18556                                                                     ; Ref: Ralf Brown's Interrupt List, INTERRUP.G
18557 000015CA C51E[1200]                         lds     bx, [ptrsav]
18558 000015CE C55F13                             lds     bx, [bx+13h]    ; [bx+IOCTL_REQ.GENERICIOCTL_PACKET]
18559 000015D1 B80381                             mov     ax, 8103h       ; ah = generic ioctl error code (81h)
18560                                                                     ; al = unknown command error (03h)
18561 000015D4 803F10                             cmp     byte [bx], 10h ; parameter buffer length = 16 bytes
18562 000015D7 7251                               jb      short gdmi_4
18563 000015D9 268A5504                           mov     dl, [es:di+4]  ; [es:di+BDS.drivenum]
18564 000015DD 885703                             mov     [bx+3], dl     ; parameter block - offset 3 - physical drive number
18565 000015E0 C6470110                           mov     byte [bx+1], 10h ; parameter block - actually used length
18566 000015E4 268B4517                           mov     ax, [es:di+17h]       ; [es:di+BDS.hiddensectors]
18567 000015E8 894708                             mov     [bx+8], ax     ; parameter block - offset 8 - partition start LBA
18568 000015EB 268B4519                           mov     ax, [es:di+19h]       ; [es:di+BDS.hiddensectors+2]
18569 000015EF 89470A                             mov     [bx+0Ah], ax   ; parameter block - offset 10
18570 000015F2 31C0                               xor     ax, ax ; 0
18571 000015F4 884702                             mov     [bx+2], al     ; drive flags = 0 (protected mode flags etc.)
18572 000015F7 89470C                             mov     [bx+0Ch], ax   ; high dword of partition start address (LBA) is 0
18573 000015FA 89470E                             mov     [bx+0Eh], ax
18574 000015FD 894704                             mov     [bx+4], ax     ; logical drive bitmap of same physical drive
18575                                                                     ; initialized as 0
18576 00001600 894706                             mov     [bx+6], ax ; 0
18577 00001603 8EC1                               mov     es, cx
18578                                             ;les    di, dword ptr es:start_bds ; 1st BDS
18579 00001605 26C43E[1901]                       les     di, [es:start_bds]
18580 0000160A B90100                             mov     cx, 1           ; bit 0 (drive A:)
18581                                   gdmi_1:
18582 0000160D 83FFFF                             cmp     di, 0FFFFh     ; last BDS ?
18583 00001610 7415                               jz      short gdmi_3   ; yes
18584 00001612 26385504                           cmp     [es:di+4], dl  ; [es:di+BDS.drivenum], dl
18585                                                                     ; is it same physical drive ?
18586 00001616 7506                               jnz     short gdmi_2   ; no
18587 00001618 094F04                             or      [bx+4], cx     ; set bit for logical drive index of this BDS
18588                                                                     ; (previously) shifted bit (which is 1/ON) is in ax:cx
18589 0000161B 094706                             or      [bx+6], ax
18590                                   gdmi_2:
18591 0000161E D1E1                               shl     cx, 1          ; shift one left for setting the next drive's bit
18592 00001620 D1D0                               rcl     ax, 1          ; set high word of the bit select (set) value
18593 00001622 26C43D                             les     di, [es:di]    ; next BDS
18594 00001625 EBE6                               jmp     short gdmi_1   ; loop until di = -1 (last BDS sign)
18595                                   gdmi_3:
18596 00001627 B80001                             mov     ax, 100h       ; success
18597                                   gdmi_4:
18598 0000162A C3                                 retn
```

```
18599
18600                                 ;--------------------------------------------------------------------------
18601
18602                                 ; 16/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
18603                                 ; 26/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
18604
18605                                 ;--------------------------------------------------------------------------
18606                                 ; MSINT13.ASM - MSDOS 6.0 - 1991
18607                                 ;--------------------------------------------------------------------------
18608                                 ; 16/03/2019 - Retro DOS v4.0
18609
18610                                 ;   int 2f function 13h allows the user to change the orig13 int_13 vector
18611                                 ;   after booting. this allows testing and implementation of custom int_13
18612                                 ;   handlers, without giving up ms-dos error recovery
18613                                 ;   entry:  ds:dx   == addr. of new int_13 handler
18614                                 ;           es:bx   == addr. of new int_13 vector used by warm boot (int19)
18615                                 ;   exit:   orig13  == address of new int_13 handler
18616                                 ;           ds:dx   == old orig13 value
18617                                 ;           es:bx   == old old13  value
18618                                 ;
18619                                 ; int 2f handler for external block drivers to communicate with the internal
18620                                 ; block driver in msdisk. the multiplex number chosen is 8. the handler
18621                                 ; sets up the pointer to the request packet in [ptrsav] and then jumps to
18622                                 ; dsk_entry, the entry point for all disk requests.
18623                                 ;
18624                                 ; on exit from this driver, we will return to the external driver
18625                                 ; that issued this int 2f, and can then remove the flags from the stack.
18626                                 ; this scheme allows us to have a small external device driver, and makes
18627                                 ; the maintainance of the various drivers (driver and msbio) much easier,
18628                                 ; since we only need to make changes in one place (most of the time).
18629                                 ;
18630                                 ;   ax=800h - check for installed handler - reserved
18631                                 ;   ax=801h - install the bds into the linked list
18632                                 ;   ax=802h - dos request
18633                                 ;   ax=803h - return bds table starting pointer in ds:di
18634                                 ;      (ems device driver hooks int 13h to handle 16kb dma overrun
18635                                 ;       problem. bds table is going to be used to get head/sector
18636                                 ;       informations without calling generic ioctl get device parm call.)
18637
18638                                 ;BIOSSEGMENT equ 70h
18639                                 DOSBIOSSEG equ 0070h ; 17/10/2022
18640
18641                                 ;;BIOSCODE:1302h (MSDOS 6.21, IO.SYS)
18642                                 ;BIOSCODE:16AAh (PCDOS 7.1, IBMBIO.COM) ; 26/12/2023
18643
18644                                 i2f_handler:                            ; here is 02C7h:1302h =      0070h:3872h
18645 0000162B 80FC13                           cmp     ah, 13h
18646 0000162E 7413                             jz      short int2f_replace_int13
18647 00001630 80FC08                           cmp     ah, 8
18648 00001633 7432                             jz      short mine
18649
18650                                 ; Check for WIN386 startup and return the BIOS instance data
18651
18652 00001635 80FC16                           cmp     ah, 16h         ; MultWin386
18653 00001638 746D                             jz      short win386call
18654 0000163A 80FC4A                           cmp     ah, 4Ah         ; multMULT
18655 0000163D 7503                             jnz     short i2f_handler_iret
18656 0000163F E99800                           jmp     handle_multmult
18657                                 ; --------------------------------------------------------------------------
18658
18659                                 i2f_handler_iret:
18660 00001642 CF                               iret
18661                                 ; --------------------------------------------------------------------------
18662
18663                                 int2f_replace_int13:
18664 00001643 FA                               cli     ; 26/12/2023
18665 00001644 50                               push    ax      ; free up a register for caller's ds
18666 00001645 8CD8                             mov     ax, ds  ; then we can use ds: -> Bios_Data
18667                                             ;;mov   ds, word [cs:0030h] ; 15/10/2022
18668                                             ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:0030h]
18669                                                             ; = [02C7h:0030h] = [0070h:25A0h]
18670 00001647 2E8E1E[3000]                     mov     ds, [cs:BIOSDATAWORD] ; 17/10/2022
18671                                           ; 19/10/2022
18672                                             ;push   word ptr ds:Orig13    ; save old value of old13 and
18673                                             ;push   word ptr ds:Orig13+2  ; orig13 so that we can
18674                                             ;push   word ptr ds:Old13     ; return them to caller
18675                                             ;push   word ptr ds:Old13+2
18676
18677                                             ; 02/09/2023 (PCDOS 7.1)
18678                                             ;push   word [Orig13]
18679 0000164C FF36[B600]                       push    word [Orig13+2]
18680                                             ;push   word [Old13]
18681 00001650 FF36[0801]                       push    word [Old13+2]
18682
18683                                             ;mov    word ptr ds:Orig13, dx; orig13 := addr. of new int_13
18684                                             ;mov    word ptr ds:Orig13+2, ax
18685                                             ;mov    word ptr ds:Old13, bx ; old13 := addr. of new boot_13
18686                                             ;mov    word ptr ds:Old13+2, es
18687
18688                                             ;mov    [Orig13], dx
18689                                             ; 02/09/2023
18690 00001654 8716[B400]                       xchg    dx, [Orig13]
18691 00001658 A3[B600]                         mov     [Orig13+2], ax
18692                                             ;mov    [Old13], bx
18693                                             ; 02/09/2023
18694 0000165B 871E[0601]                       xchg    bx, [Old13]
18695 0000165F 8C06[0801]                       mov     [Old13+2], es
18696
18697 00001663 07                               pop     es                      ; es:bx := old old13 vector
18698                                           ; 02/09/2023
18699                                             ;pop    bx
18700 00001664 1F                               pop     ds                      ; ds:dx := old orig13 vector
18701                                             ;pop    dx ; 02/09/2023
18702 00001665 58                               pop     ax
18703                                 i2f_iret:
18704 00001666 CF                               iret
18705                                 ; --------------------------------------------------------------------------
18706
18707                                 mine:
18708 00001667 3CF8                             cmp     al, 0F8h                ; iret on reserved functions
18709 00001669 73FB                             jnb     short i2f_iret
18710 0000166B 08C0                             or      al, al                  ; a get installed state request?
18711 0000166D 7503                             jnz     short disp_func
18712 0000166F B0FF                             mov     al, 0FFh
18713                                           ;jmp    short i2f_iret
18714                                           ; 02/09/2023
18715 00001671 CF                               iret
18716                                 ; --------------------------------------------------------------------------
18717
18718                                 disp_func:
18719 00001672 3C01                             cmp     al, 1                   ; request for installing bds?
18720 00001674 7418                             jz      short do_subfun_01
18721 00001676 3C03                             cmp     al, 3                   ; get bds vector?
18722 00001678 7423                             jz      short do_get_bds_vector
```

```
18723
18724                                   ; set up pointer to request packet
18725
18726 0000167A 1E                                 push    ds
18727 0000167B 2E8E1E[3000]                       mov     ds, [cs:BIOSDATAWORD] ; 17/10/2022
18728                                             ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
18729                                                             ; = [0070h:25A0h] = [02C7h:0030h]
18730                                             ; 19/10/2022
18731                                             ;mov    word ptr ds:ptrsav, bx
18732                                             ;mov    word ptr ds:ptrsav+2, es
18733 00001680 891E[1200]                         mov     [ptrsav], bx
18734 00001684 8C06[1400]                         mov     [ptrsav+2], es
18735 00001688 1F                                 pop     ds
18736                                             ;jmp    far ptr i2f_dskentry
18737                                             ; 07/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
18738                                             ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:1708h
18739 00001689 EA[5E06]7000                       jmp     DOSBIOSSEG:dsk_entry ; BIOSDATA:dsk_entry
18740                                             ;; 17/10/2022
18741                                             ;;jmp   far DOSBIOSSEG:dsk_entry
18742                                             ;jmp    DOSBIOSSEG:i2f_dskentry ; 70h:i2f_dskentry
18743                                                             ; NOTE: jump to a FAR function, not an
18744                                                             ;  IRET type function. Callers of
18745                                                             ;  this int2f subfunction will have
18746                                                             ;  to be careful to do a popf
18747
18748                                   ; ---------------------------------------------------------------------------
18749
18750                                   do_subfun_01:
18751 0000168E 06                                 push    es
18752 0000168F 1E                                 push    ds
18753 00001690 1E                                 push    ds
18754 00001691 07                                 pop     es
18755                                             ; 17/10/2022
18756 00001692 2E8E1E[3000]                       mov     ds, [cs:BIOSDATAWORD]
18757                                             ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
18758                                                             ; point ds: -> Bios_Data
18759 00001697 E8BC03                             call    install_bds
18760 0000169A 1F                                 pop     ds
18761 0000169B 07                                 pop     es
18762                                             ;jmp    short i2f_iret
18763                                             ; 02/09/2023
18764 0000169C CF                                 iret
18765                                   ; ---------------------------------------------------------------------------
18766
18767                                   do_get_bds_vector:
18768                                             ; 17/10/2022
18769 0000169D 2E8E1E[3000]                       mov     ds, [cs:BIOSDATAWORD]
18770                                             ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
18771 000016A2 C53E[1901]                         lds     di, [start_bds]
18772                                             ;lds    di, ds:start_bds
18773                                   ;ii2f_iret: ; 10/12/2022
18774                                             ;jmp    short i2f_iret
18775                                             ; 02/09/2023
18776 000016A6 CF                                 iret
18777                                   ; ---------------------------------------------------------------------------
18778
18779                                   ; 17/10/2022
18780                                   ; 16/10/2022
18781
18782                                   ; WIN386 startup stuff is done here. If starting up we set our WIN386 present
18783                                   ; flag and return instance data. If exiting, we reset the WIN386 present flag
18784                                   ; NOTE: We assume that the BIOS int 2fh is at the bottom of the chain.
18785
18786                                   win386call:
18787 000016A7 1E                                 push    ds
18788 000016A8 2E8E1E[3000]                       mov     ds, [cs:BIOSDATAWORD]
18789                                             ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
18790                                                             ; at 2C7h:30h =       70h:25A0h
18791 000016AD 3C05                               cmp     al, 5           ; Win386_Init
18792                                                             ; is itwin386 initializing?
18793 000016AF 7410                               jz      short Win386Init
18794 000016B1 3C06                               cmp     al, 6           ; Win386_Exit
18795                                                             ; is itwin386 exiting?
18796 000016B3 7523                               jnz     short win_iret ; if not, continue int2f chain
18797                                             ; 12/12/2022
18798 000016B5 F6C201                             test    dl, 1
18799                                             ;test   dx, 1           ; is itwin386 or win286 dos extender?
18800 000016B8 751E                               jnz     short win_iret ; if not win386, then continue
18801                                             ;and    ds:IsWin386, 0 ; indicate that       win386 is not present
18802 000016BA 8026[1208]00                       and     byte [IsWin386], 0
18803 000016BF EB17                               jmp     short win_iret
18804                                   ; ---------------------------------------------------------------------------
18805
18806                                   Win386Init:
18807                                             ; 12/12/2022
18808 000016C1 F6C201                             test    dl, 1
18809                                             ;test   dx, 1           ; is it win386 or win286 dos extender?
18810 000016C4 7512                               jnz     short win_iret ; if not win386, then continue
18811                                             ;or     ds:IsWin386, 1 ; Indicate WIN386 present
18812 000016C6 800E[1208]01                       or      byte [IsWin386], 1
18813                                             ;mov    word ptr ds:SI_Next, bx       ; Hook our structure into chain
18814                                             ;mov    word ptr ds:SI_Next+2, es
18815 000016CB 891E[E007]                         mov     [SI_Next], bx
18816 000016CF 8C06[E207]                         mov     [SI_Next+2], es
18817                                             ;mov    bx, offset Win386_SI ; point ES:BX to Win386_SI
18818 000016D3 BB[DE07]                           mov     bx, Win386_SI  ; 19/10/2022
18819 000016D6 1E                                 push    ds
18820 000016D7 07                                 pop     es
18821                                   win_iret:
18822 000016D8 1F                                 pop     ds
18823                                   ii2f_iret:  ; 10/12/2022
18824                                             ;jmp    short i2f_iret ; return back up the chain
18825                                             ; 02/09/2023
18826 000016D9 CF                                 iret
18827                                   ; ---------------------------------------------------------------------------
18828
18829                                   handle_multmult:
18830 000016DA 3C01                               cmp     al, 1
18831 000016DC 7514                               jnz     short try_2
18832 000016DE 1E                                 push    ds
18833 000016DF E84500                             call    HMAPtr          ; get offset of free HMA
18834                                             ; 10/12/2022
18835                                             ;xor    bx, bx
18836                                             ;dec    bx
18837 000016E2 BBFFFF                             mov     bx, 0FFFFh
18838 000016E5 8EC3                               mov     es, bx          ; seg of HMA
18839 000016E7 89FB                               mov     bx, di
18840 000016E9 F7D3                               not     bx
18841 000016EB 09DB                               or      bx, bx
18842 000016ED 7401                               jz      short try_1
18843 000016EF 43                                 inc     bx
18844                                   try_1:
18845 000016F0 1F                                 pop     ds
18846                                             ;jmp    short ii2f_iret
```

```
                                        ; 02/09/2023
000016F1 CF                             iret
                                ; ---------------------------------------------------------------------------

                                try_2:
000016F2 3C02                           cmp     al, 2           ; multMULTALLOCHMA
000016F4 7530                           jnz     short try_3
000016F6 1E                             push    ds
                                        ; 10/12/2022
                                        ;xor    di, di
                                        ;dec    di
000016F7 BFFFFF                         mov     di, 0FFFFh      ; assume not enough space
000016FA 8EC7                           mov     es, di
000016FC E82800                         call    HMAPtr          ; get offset of free HMA
000016FF 83FFFF                         cmp     di, 0FFFFh
00001702 7421                           jz      short InsuffHMA
00001704 F7DF                           neg     di              ; free space in HMA
00001706 39FB                           cmp     bx, di
00001708 7605                           jbe     short try_4
                                        ; 10/12/2022
                                        ;sub    di, di
                                        ;dec    di
0000170A BFFFFF                         mov     di, 0FFFFh
                                        ;jmp    short InsuffHMA
                                        ; 02/09/2023
0000170D 1F                             pop     ds
0000170E CF                             iret
                                ; ---------------------------------------------------------------------------

                                try_4:
                                        ;mov    di, ds:FreeHMAPtr
0000170F 8B3E[D707]                     mov     di, [FreeHMAPtr]
00001713 83C30F                         add     bx, 15
                                        ;and    bx, 0FFF0h
                                        ; 10/12/2022
00001716 80E3F0                         and     bl, 0F0h
                                        ;add    ds:FreeHMAPtr, bx ; update the free pointer
00001719 011E[D707]                     add     [FreeHMAPtr], bx
0000171D 7506                           jnz     short InsuffHMA
0000171F C706[D707]FFFF                 mov     word [FreeHMAPtr], 0FFFFh ; -1
                                        ;mov    ds:FreeHMAPtr, 0FFFFh
                                                                ; no more HMA if we have wrapped
                                InsuffHMA:
00001725 1F                             pop     ds
                                        ; 10/12/2022
                                try_3:
                                        ;jmp    short ii2f_iret
                                        ; 02/09/2023
00001726 CF                             iret
                                ; ---------------------------------------------------------------------------

                                        ; 10/12/2022
                                ;try_3:
                                        ;jmp    ii2f_iret

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

                                ; 16/10/2022

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : HMAPtr
                                ;
                                ;           Gets the offset of the free HMA area ( with respect to
                                ;                                                  seg ffff )
                                ;           If DOS has not moved high, tries to move DOS high.
                                ;           In the course of doing this, it will allocate all the HMA
                                ;           and set the FreeHMAPtr to past the end of the BIOS and
                                ;           DOS code. The call to MoveDOSIntoHMA (which is a pointer)
                                ;           enters the routine in sysinit1 called FTryToMovDOSHi.
                                ;
                                ;   RETURNS : offset of free HMA in DI
                                ;             BIOS_DATA, seg in DS
                                ;
                                ;--------------------------------------------------------------------------

                                        ; 17/10/2022
                                HMAPtr:
00001727 2E8E1E[3000]                   mov     ds, [cs:BIOSDATAWORD]
                                        ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:Bios_Data_Word]
0000172C 8B3E[D707]                     mov     di, [FreeHMAPtr]
                                        ;mov    di, ds:FreeHMAPtr
00001730 83FFFF                         cmp     di, 0FFFFh
00001733 750F                           jnz     short HMAPtr_retn
00001735 803E[DD07]00                   cmp     byte [SysinitPresent], 0
                                        ;cmp    ds:SysinitPresent, 0
0000173A 7408                           jz      short HMAPtr_retn
0000173C FF1E[D907]                     call    far [MoveDOSIntoHMA]
                                        ;call   ds:MoveDOSIntoHMA ; call far [MoveDOSIntoHMA]
00001740 8B3E[D707]                     mov     di, [FreeHMAPtr]
                                        ;mov    di, ds:FreeHMAPtr
                                HMAPtr_retn:
00001744 C3                             retn

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

                                ; 16/10/2022

                                ; move a 512 byte sector from ds:si to es:di, do not trash cx
                                ; but go ahead and update direction flag, si, & di

                                move_sector:

                                ; The 80386 microprocessor considers an access to WORD 0FFFFh in
                                ; any segment to be a fault. Theoretically, this could be handled
                                ; by the fault handler and the behavior of an 8086 could be emulated
                                ; by wrapping the high byte to offset 0000h. This would be a lot
                                ; of work and was, indeed, blown off by the Win386 guys. COMPAQ
                                ; also handles the fault incorrectly in their ROM BIOS for real
                                ; mode. Their fault handler was only designed to deal with one
                                ; special case which occurred in a magazine benchmark, but didn't
                                ; handle the general case worth beans.
                                ;
                                ; Simply changing this code to do a byte loop would work okay but
                                ; would involve a general case performance hit. Therefore, we'll
                                ; check for either source or destination offsets being within one
                                ; sector of the end of their segments and only in that case fall
                                ; back to a byte move.

00001745 FC                             cld
00001746 51                             push    cx
00001747 B90001                         mov     cx, 256
0000174A 81FE00FE                       cmp     si, 0FE00h
0000174E 770A                           ja      short movsec_bytes
```

```
18971 00001750 81FF00FE                         cmp     di, 0FE00h
18972 00001754 7704                             ja      short movsec_bytes
18973 00001756 F3A5                             rep movsw
18974 00001758 59                               pop     cx
18975 00001759 C3                               retn
18976                                   ; ---------------------------------------------------------------------------
18977
18978                                   movsec_bytes:
18979 0000175A D1E1                             shl     cx, 1
18980 0000175C F3A4                             rep movsb
18981 0000175E 59                               pop     cx
18982 0000175F C3                               retn
18983
18984                                   ; =============== S U B   R O U T I N E =======================================
18985
18986                                   ; 16/10/2022
18987
18988                                   ; check_wrap is a routine that adjusts the starting sector, starting head
18989                                   ; and starting cylinder for an int 13 request that requests i/o of a lot
18990                                   ; of sectors. it only does this for fixed disks. it is used in the sections
18991                                   ; of code that handle ecc errors and dma errors. it is necessary, because
18992                                   ; ordinarily the rom would take care of wraps around heads and cylinders,
18993                                   ; but we break down a request when we get an ecc or dma error into several
18994                                   ; i/o of one or more sectors. in this case, we may already be beyond the
18995                                   ; number of sectors on a track on the medium, and the request would fail.
18996                                   ;
18997                                   ; input conditions:
18998                                   ;   all registers set up for an int 13 request.
18999                                   ;
19000                                   ; output:
19001                                   ;   dh - contains starting head number for request
19002                                   ;   cx - contains starting sector and cylinder numbers
19003                                   ;   (the above may or may not have been changed, and are 0-based)
19004                                   ;   all other registers preserved.
19005
19006                                             ; 26/12/2023 - Retro DOS 5.0
19007                                   check_wrap:
19008 00001760 50                               push    ax
19009 00001761 53                               push    bx
19010 00001762 06                               push    es
19011 00001763 57                               push    di
19012 00001764 E86C00                           call    find_bds        ; get pointer to bds for drive in dl
19013 00001767 725E                             jb      short no_wrap   ; finished if DOS doesn't use it
19014                                           ; 26/12/2023
19015 00001769 26F6453F01                       test    byte [es:di+3Fh], 1
19016                                           ; 12/12/2022
19017                                           ;test   byte [es:di+23h], 1
19018                                           ;;test  word [es:di+23h], 1 ; [es:di+BDS.flags],fnon_removable
19019 0000176E 7457                             jz      short no_wrap   ; no wrapping for removable media
19020 00001770 268B5D13                         mov     bx, [es:di+13h]         ; [es:di+BDS.secpertrack]
19021 00001774 89C8                             mov     ax, cx
19022 00001776 83E03F                           and     ax, 3Fh         ; extract sector number
19023 00001779 39D8                             cmp     ax, bx          ; are we going to wrap?
19024 0000177B 764A                             jbe     short no_wrap
19025 0000177D F6F3                             div     bl              ; ah=new sector       #, al=# of head wraps
19026
19027                                   ; we need to be careful here. if the new sector # is 0, then we are on the
19028                                   ; last sector on that track.
19029
19030 0000177F 08E4                             or      ah, ah
19031 00001781 7503                             jnz     short not_on_bound
19032                                           ; 18/12/2022
19033 00001783 48                               dec     ax ; *
19034 00001784 88DC                             mov     ah, bl          ; set sector=BDS_BPB.BPB_SECTORSPERTRACK
19035                                                                   ; if on boundary
19036                                           ;dec    al ; *          ; also decrement # of head wraps
19037                                   not_on_bound:
19038 00001786 80E1C0                           and     cl, 0C0h        ; zero out sector #
19039 00001789 08E1                             or      cl, ah          ; or in new sector #
19040 0000178B 30E4                             xor     ah, ah          ; ax = # of head wraps
19041 0000178D 40                               inc     ax
19042 0000178E 00F0                             add     al, dh          ; add in starting head #
19043 00001790 80D400                           adc     ah, 0           ; catch any carry
19044                                           ; 02/09/2023
19045 00001793 268B5D15                         mov     bx, [es:di+15h]         ; [es:di+BDS.heads]
19046 00001797 39D8                             cmp     ax, bx
19047                                           ;cmp    ax, [es:di+15h]         ; [es:di+BDS.heads]
19048                                                                   ; are we going to wrap around a      head?
19049 00001799 7632                             jbe     short no_wrap_head ; do       not lose new head number!!
19050 0000179B 52                               push    dx              ; preserve drive number and head number
19051 0000179C 31D2                             xor     dx, dx
19052                                           ;mov    bx, [es:di+15h]         ; [es:di+BDS.heads]
19053 0000179E F7F3                             div     bx              ; dx=new head #, ax=# of cylinder wraps
19054
19055                                   ; careful here! if new head # is 0, then we are on the last head.
19056
19057 000017A0 09D2                             or      dx, dx
19058 000017A2 7507                             jnz     short no_head_bound
19059 000017A4 89DA                             mov     dx, bx          ; on boundary. set to BDS_BPB.BPB_HEADS
19060
19061                                   ; if we had some cylinder wraps, we need to reduce them by one!!
19062
19063 000017A6 09C0                             or      ax, ax
19064 000017A8 7401                             jz      short no_head_bound
19065 000017AA 48                               dec     ax              ; reduce number       of cylinder wraps
19066                                   no_head_bound:
19067 000017AB 88D7                             mov     bh, dl          ; bh has new head number
19068 000017AD 5A                               pop     dx              ; restore drive number and head number
19069 000017AE FECF                             dec     bh              ; get it 0-based
19070 000017B0 88FE                             mov     dh, bh          ; set up new head number in dh
19071 000017B2 88CF                             mov     bh, cl
19072 000017B4 80E73F                           and     bh, 3Fh         ; preserve sector number
19073 000017B7 B306                             mov     bl, 6
19074 000017B9 86D9                             xchg    cl, bl
19075 000017BB D2EB                             shr     bl, cl          ; get ms cylinder bits to ls end
19076 000017BD 00C5                             add     ch, al          ; add in cylinder wrap
19077 000017BF 10E3                             adc     bl, ah          ; add in high byte
19078 000017C1 D2E3                             shl     bl, cl          ; move up to ms       end
19079 000017C3 86CB                             xchg    bl, cl          ; restore cylinder bits       into cl
19080 000017C5 08F9                             or      cl, bh          ; or in sector number
19081                                   no_wrap:
19082 000017C7 F8                               clc
19083 000017C8 5F                               pop     di
19084 000017C9 07                               pop     es
19085 000017CA 5B                               pop     bx
19086 000017CB 58                               pop     ax
19087 000017CC C3                               retn
19088                                   ; ---------------------------------------------------------------------------
19089
19090                                   no_wrap_head:
19091 000017CD 88C6                             mov     dh, al          ; do not lose new head number
19092 000017CF FECE                             dec     dh              ; get it 0-based
19093 000017D1 EBF4                             jmp     short no_wrap
19094
```

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

                                ; 16/10/2022

                                ; this is a special version of the bds lookup code which is
                                ; based on physical drives rather than the usual logical drives
                                ; carry is set if the physical drive in dl is found, es:di -> its bds
                                ; otherwise carry is clear
                                ;
                                ; guaranteed to trash no registers except es:di

                                        ; 19/10/2022
                                find_bds:
000017D3 C43E[1901]                     les     di, [start_bds]       ; point es:di to first bds
                                fbds_1:
000017D7 26385504                       cmp     [es:di+4], dl  ; [es:di+BDS.drivenum]
000017DB 7409                           jz      short fdbs_2
000017DD 26C43D                         les     di, [es:di]    ; [es:di+BDS.link]
                                                               ; go to next bds
000017E0 83FFFF                         cmp     di, 0FFFFh
000017E3 75F2                           jnz     short fbds_1
000017E5 F9                             stc
                                fdbs_2:
000017E6 C3                             retn

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

                                ; 16/10/2022
                                        ; 17/10/2022
                                doint:
                                        ; 10/12/2022
000017E7 8A5608                         mov     dl, [bp+8]     ; [bp+INT13FRAME.olddx]
                                                               ; get physical drive number
                                        ; 19/10/2022 - Temporary !
                                        ;db     8Ah, 96h, 8, 0 ; mov dl, [bp+8]

000017EA 30E4                           xor     ah, ah
000017EC 08C0                           or      al, al
000017EE 7410                           jz      short dointdone       ; if zero sectors, return ax=0
                                        ; 10/12/2022
000017F0 8A6603                         mov     ah, [bp+3]     ; [bp+INT13FRAME.oldax+1]
                                                               ; get request code
                                        ;db     8Ah, 0A6h, 3, 0       ; mov ah, [bp+3]
000017F3 FF7610                         push    word [bp+10h]  ; [bp+INT13FRAME.oldf]
                                        ;db     0FFh, 0B6h, 10h, 0 ; push word [bp+10h]
000017F6 9D                             popf
                                        ;call   far 70h:797h ; MSDOS 6.21 IO.SYS BIOSCODE:14EAh
                                        ; 17/10/2022
000017F7 9A[0B07]7000                   call    DOSBIOSSEG:call_orig13
                                        ;;call  call_orig13    ; call far 70h:797h
                                                               ; call far KERNEL_SEGMENT:call_orig13
000017FC 9C                             pushf
                                        ; 10/12/2022
000017FD 8F4610                         pop     word [bp+10h]  ; [bp+INT13FRAME.oldf]
                                        ;db     8Fh, 86h, 10h, 0 ; pop word [bp+10h]
                                dointdone:
00001800 C3                             retn

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

                                ; 16/10/2022

                                ; this is the true int 13 handler. we parse the request to see if there is
                                ; a dma violation. if so, depending on the function, we:
                                ;   read/write break the request into three pieces and move the middle one
                                ;           into our internal buffer.
                                ;
                                ;   format      copy the format table into the buffer
                                ;   verify      point the transfer address into the buffer
                                ;
                                ; this is the biggest bogosity of all. the ibm controller does not handle
                                ; operations that cross physical 64k boundaries. in these cases, we copy
                                ; the offending sector into the buffer below and do the i/o from there.

                                ;struc INT13FRAME
                                ;.oldbp: resw
                                ;.oldax: resw
                                ;.oldbx: resw
                                ;.oldcx: resw
                                ;.olddx: resw
                                ;.oldds: resw      ; now we save caller's ds, too
                                ;.olddd: resd
                                ;.oldf:    resw
                                ;end struc

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

                                ;   entry conditions:
                                ;   ah = function
                                ;   al = number of sectors
                                ;   es:bx = dma address
                                ;   cx = packed track and sector
                                ;   dx = head and drive
                                ;   output conditions:
                                ;   no dma violation.

                                ;   use extreme caution when working with this code. In general,
                                ;     all registers are hot at all times.
                                ;
                                ;   question:  does this code handle cases where dma errors
                                ;     occur during ecc retries, and where ecc errors occur during
                                ;     dma breakdowns???? Hmmmmm.

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

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

                                        ; 26/12/2023 - Retro DOS v5.0
                                        ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:1889h
                                dtype_array:
00001801 90004000                       dd 400090h             ; 40h:90h is drive type array addr

                                ; 17/10/2022
                                ;DTYPEARRAY equ dtype_array - DOSBIOSEG_2C7h ; (14F5h for MSDOS 5.0 IO.SYS)
                                ; 09/12/2022
                                DTYPEARRAY equ dtype_array

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

                                ; stick some special stuff out of mainline

                                ; we know we're doing a format command. if we have changeline
                                ; support, then flag some special changed stuff and set changed
                                ; by format bit for all logical drives using this physical drive
```

```
19219
19220                                   format_special_stuff:
19221 00001805 803E[7700]00                       cmp     byte [fhave96], 0      ; do we have changeline support?
19222 0000180A 7459                               jz      short format_special_stuff_done ; brif not
19223 0000180C 53                                 push    bx
19224 0000180D BB4001                             mov     bx, 140h       ; fchanged_by_format+fchanged
19225 00001810 E85104                             call    set_changed_dl ; indicate that media changed by format
19226 00001813 5B                                 pop     bx
19227 00001814 EB4F                               jmp     short format_special_stuff_done
19228                                   ; ---------------------------------------------------------------------------
19229
19230                                   ; 16/10/2022
19231
19232                                   ; we know we've got ec35's on the system. Now see if we're doing
19233                                   ; a floppy. If so, create a mask and see if this particular
19234                                   ; drive is an ec35. If so, set dtype_array[drive]=93h
19235
19236                                             ; 19/10/2022
19237                                   ec35_special_stuff:
19238 00001816 84D2                               test    dl, dl         ; floppy or hard disk?
19239 00001818 7852                               js      short ec35_special_stuff_done ;      if hard drive, we're done
19240 0000181A 50                                 push    ax             ; see if this PARTICULAR drive is ec35
19241 0000181B 51                                 push    cx
19242 0000181C 88D1                               mov     cl, dl         ; turn drive number into bit map
19243 0000181E B001                               mov     al, 1          ; assume drive 0
19244 00001820 D2E0                               shl     al, cl         ; shift over correct number of times
19245 00001822 8406[A204]                         test    [ec35flag], al ; electrically compatible 3.5 incher?
19246 00001826 59                                 pop     cx
19247 00001827 58                                 pop     ax
19248 00001828 7442                               jz      short ec35_special_stuff_done
19249                                                                    ; done if this floppy is not an      ec35
19250 0000182A 53                                 push    bx             ; free up a far       pointer (es:bx)
19251 0000182B 06                                 push    es
19252                                             ; 17/10/2022
19253 0000182C 2EC41E[0118]                       les     bx, [cs:DTYPEARRAY]
19254                                             ;les    bx, dword ptr cs:DTYPEARRAY ; [cs:dtype_array]
19255                                                                    ; 0070h:3A65h =       2C7h:14F5h
19256 00001831 00D3                               add     bl, dl
19257 00001833 80D700                             adc     bh, 0          ; find entry for this drive
19258 00001836 26C60793                           mov     byte [es:bx], 93h ; establish drive type as:
19259                                                                    ; (360k disk in 360k drive,
19260                                                                    ; no double-stepping, 250 kbs transfer rate)
19261 0000183A 07                                 pop     es
19262 0000183B 5B                                 pop     bx
19263 0000183C EB2E                               jmp     short ec35_special_stuff_done
19264                                   ; ---------------------------------------------------------------------------
19265
19266                                   ; 16/10/2022
19267
19268                                   ; ps2_30 machine has some problem with ah=8h (read drive parm), int 13h.
19269                                   ; this function does not reset the common buses after the execution.
19270                                   ; to solve this problem, when we detect ah=8h, then we will save the result and
19271                                   ; will issue ah=1 (read status) call to reset the buses.
19272
19273                                   ps2_special_stuff:
19274 0000183E 803E[1E00]08                       cmp     byte [prevoper], 8 ; (ps2_30)
19275                                                                    ; read driver parm ?
19276 00001843 7407                               jz      short ps2_30_problem
19277 00001845 803E[1E00]15                       cmp     byte [prevoper], 15h
19278                                                                    ; apparently function 15h fails, too
19279 0000184A 752D                               jnz     short ps2_special_stuff_done
19280                                   ps2_30_problem:
19281 0000184C 50                                 push    ax
19282 0000184D B401                               mov     ah, 1
19283                                             ; 26/12/2023
19284                                             ;call   70h:70Bh ; PCDOS 7.1 IBMBIO.COM BIOSCODE:18D7h
19285                                             ;               ; call BIOSDATA:call_orig13
19286                                             ;;call  far 70:797h ; MSDOS 6.21 IO.SYS BIOSCODE:1543h
19287                                             ; 17/10/2022
19288 0000184F 9A[0B07]7000                       call    DOSBIOSSEG:call_orig13
19289                                             ;call   call_orig13    ; call far 70:797h
19290                                                                    ; call far KERNEL_SEGMENT:call_orig13
19291 00001854 58                                 pop     ax
19292 00001855 EB22                               jmp     short ps2_special_stuff_done
19293                                   ; ---------------------------------------------------------------------------
19294
19295                                   ; 17/10/2022
19296                                   ; 16/10/2022
19297
19298                                   ; here is the actual int13 handler
19299
19300                                   i13z:                                    ; 0070h:3ABBh =       02C7h:154Bh
19301
19302                                   ; cas -- inefficient! could push ds and load ds-> Bios_Data before
19303                                   ; vectoring up here from Bios_Data
19304
19305                                             ; 19/10/2022
19306 00001857 1E                                 push    ds             ; save caller's ds register first thing
19307                                             ;;mov   ds, word [cs:0030h]
19308                                                                    ; and set up our own ds -> Bios_Data
19309 00001858 2E8E1E[3000]                       mov     ds, [cs:BIOSDATAWORD]
19310                                             ;mov    ds, word ptr cs:BIOSDATAWORD ; [cs:0030h]
19311                                                                    ; = [02C7h:0030h] = [0070h:25A0h]
19312
19313                                   ; let the operation proceed. if there is a dma violation, then we do things
19314
19315 0000185D A3[1E00]                           mov     [prevoper], ax ; save request
19316 00001860 80FC05                             cmp     ah, 5          ; romformat
19317 00001863 74A0                               jz      short format_special_stuff
19318                                                                    ; go do special stuff for format
19319                                   format_special_stuff_done:
19320 00001865 803E[A204]00                       cmp     byte [ec35flag], 0 ; any electrically compat 3.5 inchers?
19321 0000186A 75AA                               jnz     short ec35_special_stuff
19322                                                                    ; go handle it out of line if so
19323                                   ec35_special_stuff_done:
19324                                             ; 26/12/2023
19325                                             ;call   70h:70Bh ; PCDOS 7.1 IBMBIO.COM BIOSCODE:18EDh
19326                                             ;;call  far 70:797h ; MSDOS 6.21 IO.SYS BIOSCODE:1560h
19327 0000186C 9A[0B07]7000                       call    DOSBIOSSEG:call_orig13
19328                                             ;call   call_orig13    ; call far KERNEL_SEGMENT:call_orig13
19329
19330 00001871 9C                                 pushf                  ; save result flags
19331
19332 00001872 803E[AF05]FA                       cmp     byte [model_byte], 0FAh ; is this a ps2/30?
19333                                                                    ; mdl_ps2_30
19334 00001877 74C5                               jz      short ps2_special_stuff
19335                                                                    ; exit mainline to address special
19336                                   ps2_special_stuff_done:                  ; ps2/30 problem if so
19337 00001879 9D                                 popf
19338 0000187A 7221                               jb      short goterr13 ; error on original orig13 call-thru?
19339                                   ret_from_i13:
19340 0000187C 1F                                 pop     ds
19341 0000187D CA0200                             retf    2              ; restore ds & iret w/flags
19342                                   ; ---------------------------------------------------------------------------
```

```
19343
19344                                   ; most of our code exits through here. If carry isn't set, then
19345                                   ; just do a simple exit. Else doublecheck that we aren't getting
19346                                   ; a changeline error.
19347
19348                                   i13ret_ck_chglinerr:
19349 00001880 73FA                              jnb     short ret_from_i13 ; done if not an error termination
19350                                   i13_ret_error:
19351 00001882 80FC06                           cmp     ah, 6           ; did i see a change event?
19352 00001885 7513                             jnz     short int13b    ; skip if wrong       error
19353 00001887 08D2                             or      dl, dl          ; is this for the hard disk?
19354 00001889 780F                             js      short int13b    ; yes, ignore
19355 0000188B 803E[7700]00                     cmp     byte [fhave96], 0
19356 00001890 7408                             jz      short int13b    ; just in case ROM returned this
19357                                                                   ; error even though it told us it
19358                                                                   ; never would
19359 00001892 53                               push    bx
19360 00001893 BB4000                           mov     bx, 40h         ; fchanged
19361 00001896 E8CB03                           call    set_changed_dl
19362 00001899 5B                               pop     bx
19363                                   int13b:
19364 0000189A F9                               stc                     ; now return the error
19365 0000189B EBDF                             jmp     short ret_from_i13
19366                                   ; ---------------------------------------------------------------------------
19367
19368                                   ; some kind of error occurred. see if it is dma violation
19369
19370                                   goterr13:
19371 0000189D 80FC09                           cmp     ah, 9           ; dma error?
19372 000018A0 747C                             jz      short gotdmaerr
19373                                   goterr13_xxxx:
19374 000018A2 80FC11                           cmp     ah, 11h         ; ecc error?
19375 000018A5 75DB                             jnz     short i13_ret_error ; other error. just     return back.
19376 000018A7 803E[A905]01                     cmp     byte [media_set_for_format], 1 ; formatting?
19377 000018AC 74D4                             jz      short i13_ret_error
19378
19379 000018AE 803E[1F00]02                     cmp     byte [prevoper+1], 2
19380                                           ;cmp    byte ptr ds:prevoper+1,      2 ; ecc-corrected error
19381                                                                   ; (2 = romread)
19382                                                                   ; ECC correction only applies to reads
19383 000018B3 75CD                             jnz     short i13_ret_error
19384
19385 000018B5 30E4                             xor     ah, ah
19386                                           ;;call  far 70:797h ; MSDOS 6.21 IO.SYS BIOSCODE:15ABh
19387                                           ; 17/10/2022
19388 000018B7 9A[0B07]7000                     call    DOSBIOSSEG:call_orig13
19389                                           ;call   call_orig13     ; call far KERNEL_SEGMENT:call_orig13
19390                                                                   ; call far 70:797h
19391 000018BC A1[1E00]                         mov     ax, [prevoper]
19392 000018BF 30E4                             xor     ah, ah          ; return code =       no error
19393 000018C1 3C01                             cmp     al, 1           ; if request for one sector, assume ok
19394 000018C3 74B7                             jz      short ret_from_i13 ; return with carry clear
19395 000018C5 53                               push    bx
19396 000018C6 51                               push    cx
19397 000018C7 52                               push    dx
19398 000018C8 A2[2000]                         mov     [number_of_sec], al
19399                                   loop_ecc:
19400 000018CB B80102                           mov     ax, 201h        ; read one sector
19401
19402                                   ; we do reads one sector at a time. this ensures that we will eventually
19403                                   ; finish the request since ecc errors on one sector do read in that sector.
19404                                   ;
19405                                   ; we need to put in some "intelligence" into the ecc handler to handle reads
19406                                   ; that attempt to read more sectors than are available on a particular
19407                                   ; track.
19408                                   ;
19409                                   ; we call check_wrap to set up the sector #, head # and cylinder # for
19410                                   ; this request.
19411                                   ;
19412                                   ; at this point, all registers are set up for the call to orig13, except
19413                                   ; that there may be a starting sector number that is bigger than the number
19414                                   ; of sectors on a track.
19415                                   ;
19416 000018CE E88FFE                           call    check_wrap      ; get correct parameters for int 13
19417                                           ;;call  far 70:797h ; MSDOS 6.21 IO.SYS BIOSCODE:15C5h
19418                                           ; 17/10/2022
19419 000018D1 9A[0B07]7000                     call    DOSBIOSSEG:call_orig13
19420                                           ;call   call_orig13     ; call far KERNEL_SEGMENT:call_orig13
19421 000018D6 730C                             jnb     short ok11_op
19422 000018D8 80FC09                           cmp     ah, 9           ; DMA error during ECC read?
19423 000018DB 741B                             jz      short handle_dma_during_ecc
19424 000018DD 80FC11                           cmp     ah, 11h         ; only allow ecc errors
19425 000018E0 7510                             jnz     short ok11_exit_err
19426                                           ; 10/12/2022
19427                                           ; xor ax ax -> ah = 0
19428                                           ;mov    ah, 0           ; ecc error. reset the system again.
19429 000018E2 31C0                             xor     ax, ax          ; clear the error code so that if this
19430                                                                   ; was the last sector, no error      code
19431                                                                   ; will be returned for the corrected
19432                                                                   ; read. (clear carry too.)
19433                                   ok11_op:
19434 000018E4 FE0E[2000]                               dec     byte [number_of_sec]
19435 000018E8 7409                             jz      short ok11_exit       ; all done?
19436 000018EA FEC1                             inc     cl              ; advance sector number
19437                                                                   ; add 200h to address
19438 000018EC FEC7                             inc     bh
19439 000018EE FEC7                             inc     bh
19440 000018F0 EBD9                             jmp     short loop_ecc
19441                                   ; ---------------------------------------------------------------------------
19442
19443                                   ; locate error returns centrally
19444
19445                                   ok11_exit_err:
19446 000018F2 F9                               stc                     ; set carry bit again.
19447                                   ok11_exit:
19448 000018F3 5A                               pop     dx
19449 000018F4 59                               pop     cx
19450 000018F5 5B                               pop     bx
19451 000018F6 EB88                             jmp     short i13ret_ck_chglinerr
19452                                   ; ---------------------------------------------------------------------------
19453
19454                                   ; do the single sector read again, this time into our temporary
19455                                   ; buffer, which is guaranteed not to have a DMA error, then
19456                                   ; move the data to its proper location and proceed
19457
19458                                   handle_dma_during_ecc:
19459 000018F8 06                               push    es
19460 000018F9 53                               push    bx
19461 000018FA BB[5201]                         mov     bx, disksector
19462 000018FD 1E                               push    ds
19463 000018FE 07                               pop     es              ; point es:bx to buffer
19464 000018FF B80102                           mov     ax, 201h        ; read one sector
19465                                           ;;call  far 70:797h ; MSDOS 6.21 IO.SYS BIOSCODE:15F8h
19466                                           ; 17/10/2022
```

```
19467 00001902 9A[0B07]7000                     call   DOSBIOSSEG:call_orig13
19468                                           ;call  call_orig13    ; call far KERNEL_SEGMENT:call_orig13
19469 00001907 5B                               pop    bx
19470 00001908 07                               pop    es
19471 00001909 7305                             jnb    short handle_dma_during_ecc_noerr
19472 0000190B 80FC11                           cmp    ah, 11h
19473 0000190E 75E2                             jnz    short ok11_exit_err ; if anything but ecc error, bomb out
19474
19475                                   ; now we're kosher. Copy the data to where it belongs and resume
19476                                   ; the ECC looping code.
19477
19478                                   handle_dma_during_ecc_noerr:
19479 00001910 56                               push   si
19480 00001911 57                               push   di
19481 00001912 89DF                             mov    di, bx
19482 00001914 BE[5201]                         mov    si, disksector
19483 00001917 E82BFE                           call   move_sector
19484 0000191A 5F                               pop    di
19485 0000191B 5E                               pop    si
19486 0000191C EBC6                             jmp    short ok11_op
19487                                   ; ---------------------------------------------------------------------------
19488
19489                                   ; we truly have a dma violation. restore register ax and retry the
19490                                   ; operation as best we can.
19491
19492                                   gotdmaerr:
19493 0000191E A1[1E00]                         mov    ax, [prevoper] ; 19/10/2022
19494 00001921 FB                               sti
19495 00001922 80FC02                           cmp    ah, 2          ; romread
19496 00001925 723B                             jb     short i13_done_dmaerr
19497                                                                  ; just pass dma error thru for
19498                                                                  ; functions we don't handle
19499 00001927 80FC04                           cmp    ah, 4          ; romverify
19500 0000192A 743C                             jz     short intverify
19501 0000192C 80FC05                           cmp    ah, 5          ; romformat
19502 0000192F 7448                             jz     short intformat
19503 00001931 772F                             ja     short i13_done_dmaerr
19504
19505                                   ; we are doing a read/write call. check for dma problems
19506
19507                                   ;   ******** set up stack frame here!!! ********
19508
19509 00001933 52                                push   dx
19510 00001934 51                               push   cx
19511 00001935 53                               push   bx
19512 00001936 50                               push   ax
19513 00001937 55                               push   bp
19514 00001938 89E5                             mov    bp, sp
19515 0000193A 8CC2                             mov    dx, es         ; check for 64k boundary error
19516                                           ; 26/12/2023
19517                                           ;add   dx, dx
19518                                           ;add   dx, dx
19519                                           ;add   dx, dx
19520                                           ;add   dx, dx         ; dx = dx*16
19521 0000193C D1E2                             shl    dx, 1
19522 0000193E D1E2                             shl    dx, 1
19523 00001940 D1E2                             shl    dx, 1
19524 00001942 D1E2                             shl    dx, 1          ; segment converted to absolute      address
19525 00001944 01DA                             add    dx, bx         ; combine with offset
19526 00001946 81C2FF01                         add    dx, 511        ; simulate a transfer
19527
19528                                   ; if carry is set, then we are within 512 bytes of the end of the segment.
19529                                   ; we skip the first transfer and perform the remaining buffering and transfer
19530
19531 0000194A 7303                              jnb    short no_skip_first
19532 0000194C E98300                           jmp    bufferx        ; restore dh=head & do buffer
19533                                   ; ---------------------------------------------------------------------------
19534
19535                                   no_skip_first:
19536 0000194F D0EE                             shr    dh, 1          ; dh = number of sectors before      address
19537 00001951 B480                             mov    ah, 128        ; ah = max number of sectors in      segment
19538 00001953 28F4                             sub    ah, dh
19539
19540                                   ; ah is now the number of sectors that we can successfully write in this
19541                                   ; segment. if this number is above or equal to the requested number, then we
19542                                   ; continue the operation as normal. otherwise, we break it into pieces.
19543                                   ;
19544                                   ; wait a sec. this is goofy. the whole reason we got here in the
19545                                   ; first place is because we got a dma error. so it's impossible
19546                                   ; for the whole block to fit, unless the dma error was returned
19547                                   ; in error.
19548
19549 00001955 38C4                             cmp    ah, al         ; can we fit it       in?
19550 00001957 7236                             jb     short doblock  ; no, perform blocking.
19551
19552                                   ; yes, the request fits. let it happen.
19553
19554 00001959 8A7609                            mov    dh, [bp+9]     ; [bp+INT13FRAME.olddx+1]
19555                                                                  ; set up head number
19556 0000195C E888FE                           call   doint
19557 0000195F E9D900                           jmp    bad13          ; and return from this place
19558                                   ; ---------------------------------------------------------------------------
19559
19560                                   i13_done_dmaerr:
19561 00001962 B409                             mov    ah, 9          ; pass dma error thru to caller
19562 00001964 F9                               stc
19563 00001965 E914FF                           jmp    ret_from_i13   ; return with error,
19564                                                                  ; we know it's not a changeline error
19565                                   ; ---------------------------------------------------------------------------
19566
19567                                   ; verify the given sectors. place the buffer pointer into our space.
19568
19569                                   intverify:
19570 00001968 06                               push   es             ; save caller's dma address
19571 00001969 53                               push   bx
19572 0000196A 1E                               push   ds             ; es:bx -> Bios_Data:disksector
19573 0000196B 07                               pop    es
19574                                   dosimple:
19575 0000196C BB[5201]                         mov    bx, disksector
19576                                                                  ; do the i/o from Bios_Data:disksector
19577                                           ;;call far 70:797h ; MSDOS 6.21 IO.SYS BIOSCODE:1665h
19578                                           ; 17/10/2022
19579 0000196F 9A[0B07]7000                     call   DOSBIOSSEG:call_orig13
19580                                           ;call  call_orig13    ; call far KERNEL_SEGMENT:call_orig13
19581 00001974 5B                               pop    bx
19582 00001975 07                               pop    es
19583 00001976 E907FF                           jmp    i13ret_ck_chglinerr
19584                                   ; ---------------------------------------------------------------------------
19585
19586                                   ; format operation. copy the parameter table into Bios_Data:disksector
19587
19588                                   intformat:
19589 00001979 06                               push   es
19590 0000197A 53                               push   bx
```

```
19591 0000197B 56                              push    si
19592 0000197C 57                              push    di
19593 0000197D 1E                              push    ds
19594
19595                                  ; point ds to the caller's dma buffer, es to Bios_Data
19596                                  ; in other words, swap (ds, es)
19597
19598 0000197E 06                              push    es
19599 0000197F 1E                              push    ds
19600 00001980 07                              pop     es
19601 00001981 1F                              pop     ds
19602 00001982 89DE                            mov     si, bx
19603 00001984 BF[5201]                        mov     di, disksector
19604 00001987 E8BBFD                          call    move_sector    ; user's data into Bios_Data:disksector
19605 0000198A 1F                              pop     ds
19606 0000198B 5F                              pop     di
19607 0000198C 5E                              pop     si             ; do the i/o from
19608 0000198D EBDD                            jmp     short dosimple ; Bios_Data:disksector
19609                                  ; ---------------------------------------------------------------------------
19610
19611                                  ; we can't fit the request into the entire block. perform the operation on
19612                                  ; the first block.
19613                                  ;
19614                                  ; doblock is modified to correctly handle multi-sector disk i/o.
19615                                  ; old doblock had added the number of sectors i/oed (ah in old doblock) after
19616                                  ; the doint call to cl. observing only the lower 6 bits of cl(=max. 64) can
19617                                  ; represent a starting sector, if ah was big, then cl would be clobbered.
19618                                  ; by the way, we still are going to use cl for this purpose since checkwrap
19619                                  ; routine will use it as an input. to prevent cl from being clobbered, a
19620                                  ; safe number of sectors should be calculated like "63 - # of sectors/track".
19621                                  ; doblock will handle the first block of requested sectors within the
19622                                  ; boundary of this safe value.
19623
19624                                            ; 26/12/2023 - Retro DOS v5.0
19625                                  doblock:
19626
19627                                  ; try to get the # of sectors/track from bds via rom drive number.
19628                                  ; for any mini disks installed, here we have to pray that they have the
19629                                  ; same # of sector/track as the main dos partition disk drive.
19630
19631 0000198F 8B5608                            mov     dx, [bp+8]     ; [bp+INT13FRAME.olddx]
19632                                                                   ; get head #, drive #
19633 00001992 51                              push    cx
19634 00001993 06                              push    es
19635 00001994 57                              push    di             ; ah - # of sectors before dma boundary
19636                                                                   ; al - requested # of sectors for i/o.
19637 00001995 E83BFE                          call    find_bds
19638 00001998 268B4D13                        mov     cx, [es:di+13h]       ; [es:di+BDS.secpertrack]
19639                                          ; 26/12/2023
19640 0000199C 26F6453F01                      test    byte [es:di+3Fh], 1
19641                                          ; 12/12/2022
19642                                          ;test  byte [es:di+23h], 1
19643                                          ;;test word [es:di+23h], 1 ; [es:di+BDS.flags],fnon_removable
19644 000019A1 5F                              pop     di
19645 000019A2 07                              pop     es
19646 000019A3 88E0                            mov     al, ah         ; set al=ah for       floppies
19647 000019A5 7404                            jz      short doblockflop ; they are track by track operation
19648 000019A7 B43F                            mov     ah, 63         ; ah = 63-secpt       (# safe sectors??)
19649 000019A9 28CC                            sub     ah, cl         ; al - # of sectors before dma boundary
19650                                  doblockflop:
19651 000019AB 59                              pop     cx
19652                                  doblockcontinue:
19653 000019AC 38C4                            cmp     ah, al         ; if safe_# >= #_of_sectors_to_go_before dma,
19654 000019AE 7305                            jnb     short doblocklast ; then #_of_sectors_to_go as it is for doint.
19655 000019B0 50                              push    ax
19656 000019B1 88E0                            mov     al, ah         ; otherwise, set al to ah to operate.
19657 000019B3 EB03                            jmp     short doblockdoint
19658                                  ; ---------------------------------------------------------------------------
19659
19660                                  doblocklast:
19661 000019B5 88C4                            mov     ah, al
19662 000019B7 50                              push    ax
19663                                  doblockdoint:                         ; let ah = al =       # of sectors for this shot
19664 000019B8 E82CFE                          call    doint
19665 000019BB 727E                            jb      short bad13    ; something happened, bye!
19666 000019BD 58                              pop     ax
19667 000019BE 286602                          sub     [bp+2], ah     ; sub [bp+INT13FRAME.oldax], ah
19668                                                                   ; decrement by the successful operation
19669 000019C1 00E1                            add     cl, ah         ; advance sector #. safety gauranteed.
19670 000019C3 00E7                            add     bh, ah         ; advance dma addres
19671 000019C5 00E7                            add     bh, ah         ; twice for 512 byte sectors
19672 000019C7 38C4                            cmp     ah, al         ; check the previous value
19673 000019C9 740A                            jz      short buffer   ; if #_of_sectors_to_go       < safe_#,
19674                                                                   ; then we are done already.
19675 000019CB 28E0                            sub     al, ah         ; otherwise,
19676                                                                   ; #_sector_to_go = #_of_sector_to_go - safe_#
19677 000019CD E890FD                          call    check_wrap     ; get new cx, dh for the next operation.
19678 000019D0 EBDA                            jmp     short doblockcontinue ;       handles next sectors left.
19679                                  ; ---------------------------------------------------------------------------
19680
19681                                  bufferx:
19682 000019D2 8A7609                          mov     dh, [bp+9]     ; [bp+INT13FRAME.olddx+1]
19683                                                                   ; set up head number
19684                                  buffer:
19685 000019D5 53                              push    bx
19686 000019D6 8A6603                          mov     ah, [bp+3]     ; [bp+INT13FRAME.oldax+1]
19687 000019D9 80FC03                          cmp     ah, 3          ; romwrite
19688 000019DC 7525                            jnz     short doread   ;
19689
19690                                  ; copy the offending sector into local buffer
19691
19692 000019DE 06                               push    es
19693 000019DF 1E                              push    ds
19694 000019E0 56                              push    si
19695 000019E1 57                              push    di
19696 000019E2 1E                              push    ds             ; exchange segment registers
19697 000019E3 06                              push    es
19698 000019E4 1F                              pop     ds
19699 000019E5 07                              pop     es
19700 000019E6 BF[5201]                        mov     di, disksector ; where to move
19701 000019E9 57                              push    di             ; save it
19702 000019EA 89DE                            mov     si, bx         ; source
19703 000019EC E856FD                          call    move_sector    ; move sector into local buffer
19704 000019EF 5B                              pop     bx             ; new transfer address
19705                                                                 ; (es:bx = Bios_Data:diskbuffer)
19706 000019F0 5F                              pop     di             ; restore caller's di & si
19707 000019F1 5E                              pop     si
19708 000019F2 1F                              pop     ds             ; restore Bios_Data
19709
19710                                  ; see if we are wrapping around a track or head
19711
19712 000019F3 B001                             mov     al, 1          ; [bp+INT13FRAME.olddx]
19713                                                                   ; get drive number
19714 000019F5 8A5608                          mov     dl, [bp+8]
```

```
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
000019FB E8E9FD                         call    doint
000019FE 07                             pop     es              ; restore caller's dma segment
000019FF 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
00001A5C 268A4504                       mov     al, [es:di+4]  ; [es:di+BDS.drivenum]
00001A60 384404                         cmp     [si+4], al     ; does this one       share a physical
                                                        ; drive with new one?
00001A63 7518                           jnz     short next_bds
00001A65 B310                           mov     bl, 10h        ; fi_am_mult
                                        ; 26/12/2023
00001A67 26085D3F                       or      [es:di+3Fh], bl
                                        ;or     [es:di+23h], bl       ; [es:di+BDS.flags]
                                                        ; set both of them to i_am_mult      if so
00001A6B 085C3F                         or      [si+3Fh], bl
                                        ;or     [si+23h], bl   ; [si+BDS.flags]
00001A6E 2680653FDF                     and     byte [es:di+3Fh], 0DFh
                                        ;and    byte [es:di+23h], 0DFh ; [es:di+BDS.flags],~fi_own_physical
                                                        ; we don't own it
00001A73 8A5C3F                         mov     bl, [si+3Fh]
                                        ;mov    bl, [si+23h]   ; [si+BDS.flags]
                                                        ; determine if changeline available
00001A76 80E302                         and     bl, 2          ; fchangeline
00001A79 26085D3F                       or      [es:di+3Fh], bl
                                        ;or     [es:di+23h], bl       ; [es:di+BDS.flags]
                                    next_bds:
                                        ; 02/09/2023 (PCDOS 7.1)
00001A7D B8FFFF                         mov     ax, 0FFFFh     ; -1
00001A80 3904                           cmp     [si], ax       ; [si+BDS.link],-1
                                        ;cmp    word [si], 0FFFFh ; [si+BDS.link],-1
                                                        ; are we at end       of list?
00001A82 75D6                           jnz     short loop_next_bds
00001A84 8C4402                         mov     [si+2], es     ; [si+BDS.link+2],es
                                                        ; install bds
00001A87 893C                           mov     [si], di
00001A89 268905                         mov     [es:di], ax    ; [es:di+BDS.link],-1
                                        ;mov    word [es:di], 0FFFFh ; [es:di+BDS.link],-1
                                                        ; set next pointer to null
00001A8C 1F                             pop     ds

                                    ; 01/07/2023 - Retro DOS v4.2 (MSDOS 6.21 IO.SYS - BIOSCODE:1785h)
                                    ; 16/10/2022 (MSDOS 6.0 Code)

                                    ; **** If the new drive has a higher EOT value, we must alter the
                                    ;      'eot' variable appropriately.

                                            ; 26/12/2023
00001A8D 268A4550                           mov     al, [es:di+50h]       ; [es:di+BDS.rsecpertrack]
                                            ; 01/06/2019
                                            ;mov    al,[es:di+52]
                                            ; 22/07/2023
                                            ;mov    al,[es:di+BDS.rsecpertrack]
00001A91 3A06[2C01]                         cmp     al,[eot]
00001A95 7603                               jbe     short _eot_ok
00001A97 A2[2C01]                           mov     [eot],al
                                    _eot_ok:
00001A9A C3                                 retn

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

                                    ; 17/10/2022
                                    ;DRVLET     equ drvlet - DOSBIOSEG_2C7h
                                    ;SNGMSG     equ sngmsg - DOSBIOSEG_2C7h
                                    ; 09/12/2022
                                    DRVLET equ drvlet
                                    SNGMSG equ sngmsg

                                    ; 16/10/2022

                                    ;--------------------------------------------------------------------------
                                    ;  ask to swap the disk in drive a:
                                    ;   es:di -> bds
                                    ;   ds -> Bios_Data
                                    ;--------------------------------------------------------------------------

                                            ; 26/12/2023 - Retro DOS v5.0

                                            ; 19/10/2022
00001A9B F606[1208]01               swpdsk:             test   byte [IsWin386], 1
                                            ;test   ds:IsWin386, 1 ; Is win386 present?
00001AA0 7405                               jz      short no_win386       ; no, skip SetFocus

                                            ; set focus to the correct VM
                                            ;call   far ptr 70h:813h ; PCDOS 7.1 IBMBIO.COM BIOSCODE:1B2Ch
                                            ;;call  far 70h:8D1h   ; MSDOS 6.21 IO.SYS BIOSCODE:179Ah
                                            ; 17/10/2022
00001AA2 9A[1308]7000                       call    DOSBIOSSEG:V86_Crit_SetFocus ; BIOSDATA:V86_Crit_SetFocus
                                            ;call   far ptr V86_Crit_SetFocus ; call far 70h:8D1h
                                                                ; call far KERNEL_SEGMENT:V86_Crit_SetFocus
                                    no_win386:
00001AA7 51                                 push    cx
00001AA8 52                                 push    dx
00001AA9 268A5505                           mov     dl, [es:di+5]  ; [es:di+BDS.drivelet]
                                                        ; get the drive       letter

                                    ; WARNING : next two instructions assume that if the new disk is for drive B
                                    ;           then existing dsk is drive A & vice versa

00001AAD 88D6                               mov     dh, dl
00001AAF 80F601                             xor     dh, 1
00001AB2 29C9                               sub     cx, cx         ; nobody has handled swap disk
00001AB4 B8004A                             mov     ax, 4A00h      ; multMULT<<8)|multMULTSWPDSK
                                                        ; broadcast code for swap disk
                                                        ; Broadcast it
00001AB7 CD2F                               int     2Fh
00001AB9 41                                 inc     cx             ; cx == -1 ?
00001ABA 741E                               jz      short swpdsk9  ; somebody has handled it

                                    ; using a different drive in a one drive system so request the user change disks

00001ABC 80C241                             add     dl, 'A'
                                            ; 17/10/2022
00001ABF 2E8816[F91A]                       mov     [cs:DRVLET], dl       ; "A: and press any key when ready\r\n\n"
                                            ; 16/10/2022
                                            ;;mov   byte [cs:drvlet], dl
                                            ;mov    byte ptr cs:17E4h, dl ; [cs:drvlet]
                                                        ; 0070h:3D54h =       2C7h:17E4h
00001AC4 BE[DD1A]                           mov     si, SNGMSG     ; "\r\nInsert diskette for drive "
                                            ;mov    si, 17C8h      ; sngmsg
                                                        ; 0070h:3D38h =       2C7h:17C8h
00001AC7 53                                 push    bx
00001AC8 2E                                 cs
00001AC9 AC                                 lodsb                  ; get the next character of the message
                                            ;lods   byte ptr cs:[si]
                                    wrmsg_loop:
00001ACA CD29                               int     29h            ; DOS 2+ internal - FAST PUTCHAR
                                                        ; AL = character to display
00001ACC 2E                                 cs
```

```
19963 00001ACD AC                               lodsb
19964                                           ;lods   byte ptr cs:[si] ; cs lodsb
19965                                                                    ; get the next character of the      message
19966 00001ACE 08C0                             or      al, al
19967 00001AD0 75F8                             jnz     short wrmsg_loop
19968 00001AD2 E833E7                           call    con_flush       ; flush out keyboard queue
19969                                                                    ; call rom-bios
19970 00001AD5 30E4                             xor     ah, ah
19971 00001AD7 CD16                             int     16h             ; KEYBOARD - READ CHAR FROM BUFFER, WAIT IF EMPTY
19972                                                                    ; Return: AH = scan code, AL = character
19973 00001AD9 5B                               pop     bx
19974                                   swpdsk9:
19975 00001ADA 5A                               pop     dx
19976 00001ADB 59                               pop     cx
19977 00001ADC C3                               retn
19978
19979                                   ; ---------------------------------------------------------------------------
19980
19981                                   ; 16/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
19982
19983                                   ;--------------------------------------------------------
19984                                   ; include msbio.cl2 (MSDOS 6.0, 1991)
19985                                   ;--------------------------------------------------------
19986                                   ; (MSDOS 6.21 IO.SYS BIOSCODE:17D5h)
19987                                   ;--------------------------------------------------------
19988                                   ; 17/03/2019 - Retro DOS v4.0
19989                                   ; 26/12/2023 - Retro DOS v5.0
19990
19991                                               ; MSDOS 5.0 IO.SYS offset 0070h:3D38h or 02C7h:17C8h
19992                                               ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:1B67h
19993 00001ADD 0D0A                     sngmsg:             db 0Dh,0Ah
19994 00001ADF 496E73657274206469-                  db 'Insert diskette for drive '
19994 00001AE8 736B6574746520666F-
19994 00001AF1 7220647269766520
19995
19996                                               ; MSDOS 5.0 IO.SYS offset 0070h:3D54h or 02C7h:17E4h
19997                                               ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:1B83h
19998 00001AF9 413A20616E64207072-      drvlet:             db 'A: and press any key when ready',0Dh,0Ah
19998 00001B02 65737320616E79206B-
19998 00001B0B 6579207768656E2072-
19998 00001B14 656164790D0A
19999 00001B1A 0A00                                 db 0Ah,0
20000
20001                                   ; =============== S U B   R O U T I N E =======================================
20002
20003                                   ;---------------------------------------------------------------------------
20004                                   ; input : es:di points to current bds for drive.
20005                                   ; return : zero set if no open files
20006                                   ;      zero reset if open files
20007                                   ;---------------------------------------------------------------------------
20008
20009                                               ; 26/12/2023 - Retro DOS v5.0
20010                                   chkopcnt:
20011 00001B1C 26837D3C00                           cmp     word [es:di+3Ch], 0
20012                                               ;cmp    word [es:di+20h], 0 ; [es:di+BDS.opcnt]
20013 00001B21 C3                                   retn
20014
20015                                   ; =============== S U B   R O U T I N E =======================================
20016
20017                                   ;---------------------------------------------------------------------------
20018                                   ; at media check time, we need to really get down and check what the change is.
20019                                   ; this is guaranteed to be expensive.
20020                                   ;
20021                                   ;   es:di -> bds, ds -> Bios_Data
20022                                   ;---------------------------------------------------------------------------
20023
20024                                               ; 26/12/2023 - Retro DOS v5.0
20025                                               ; PCDOS 7.1 IBMBIO.COM - BIOSCODE:1BA6h
20026                                   mediacheck:
20027 00001B22 E852EE                               call    checksingle    ; make sure correct disk is in place
20028 00001B25 31F6                                 xor     si, si
20029 00001B27 E86101                               call    haschange
20030 00001B2A 742F                                 jz      short mediaret
20031                                               ; 26/12/2023
20032                                               ;test   byte [es:di+3Fh], 40h ; [es:di+BDS.flags], fchanged ; 40h
20033 00001B2C E85001                               call    checkromchange
20034 00001B2F 752B                                 jnz     short mediadovolid
20035 00001B31 50                                   push    ax
20036 00001B32 52                                   push    dx
20037 00001B33 268A5504                             mov     dl, [es:di+4]  ; [es:di+BDS.drivenum]
20038                                                                      ; set logical drive number
20039 00001B37 B416                                 mov     ah, 16h
20040 00001B39 CD13                                 int     13h             ; DISK - FLOPPY       DISK - CHANGE OF DISK STATUS (AT,XT2,XT286,CONV,PS)
20041                                                                      ; DL = drive to       check
20042                                                                      ; Return: AH = disk change status
20043 00001B3B 5A                                   pop     dx
20044 00001B3C 58                                   pop     ax
20045 00001B3D 721D                                 jb      short mediadovolid
20046 00001B3F BE0100                               mov     si, 1          ; signal no change
20047
20048                                   ; there are some drives with changeline that "lose" the changeline indication
20049                                   ; if a different drive is accessed after the current one. in order to avoid
20050                                   ; missing a media change, we return an "i don't know" to dos if the changeline
20051                                   ; is not active and we are accessing a different drive from the last one.
20052                                   ; if we are accessing the same drive, then we can safely rely on the changeline
20053                                   ; status.
20054                                               ; 19/10/2022
20055 00001B42 8A1E[1E01]                           mov     bl, [tim_drv]  ; get last drive accessed
20056 00001B46 26385D04                             cmp     [es:di+4], bl  ; [es:di+BDS.drivenum]
20057                                                                      ; (If the last drive accessed is not current drive
20058                                                                      ; media change status may be incorrect.      So,
20059                                                                      ; "I don't now" will be returned even if it is indicated
20060                                                                      ; as media is not changed.)
20061 00001B4A 740F                                 jz      short mediaret ; (same drive,
20062                                                                      ; media changeline indication is reliable)
20063
20064                                   ; do the 2 second twiddle. if time >= 2 seconds, do a volid check.
20065                                   ; otherwise return "i don't know" (strictly speaking, we should return a
20066                                   ; "not changed" here since the 2 second test said no change.)
20067
20068 00001B4C 50                                   push    ax
20069 00001B4D 51                                   push    cx
20070 00001B4E 52                                   push    dx
20071 00001B4F E8D8EA                               call    Check_Time_Of_Access
20072 00001B52 5A                                   pop     dx
20073 00001B53 59                                   pop     cx
20074 00001B54 58                                   pop     ax
20075 00001B55 09F6                                 or      si, si
20076 00001B57 7403                                 jz      short mediadovolid ; check_time      says ">= 2 secs       passed"
20077                                                                      ; (volume id will be checked)
20078 00001B59 31F6                                 xor     si, si         ; return "i don't know"
20079                                   mediaret:
20080 00001B5B C3                                   retn
20081                                   ; ---------------------------------------------------------------------------
```

```
20082
20083                                   ; somehow the media was changed. look at vid to see. we do not look at fat
20084                                   ; because this may be different since we only set medbyt when doing a read
20085                                   ; or write.
20086
20087                                   mediadovolid:
20088 00001B5C E877EB                             call    GetBp           ; build a new bpb in current bds
20089 00001B5F 72FA                               jb      short mediaret
20090 00001B61 E82D00                             call    check_vid
20091 00001B64 73F5                               jnb     short mediaret
20092 00001B66 E940F2                             jmp     maperror        ; fix up al for       return to dos
20093                                   ; ---------------------------------------------------------------------------
20094
20095                                   ; simple, quick check of latched change. if no indication, then return
20096                                   ; otherwise do expensive check. if the expensive test fails, pop off the
20097                                   ; return and set al = 15 (for invalid media change) which will be returned to
20098                                   ; dos.
20099                                   ;
20100                                   ; for dos 3.3, this will work only for the drive that has changeline.
20101
20102                                   ;   call with es:di -> bds, ds -> Bios_Data
20103                                   ;   ***** warning:  this routine will return one level up on the stack
20104                                   ;                   if an error occurs!
20105
20106                                   checklatchio:
20107
20108                                   ; if returning fake bpb then assume the disk has not changed
20109
20110                                             ; 26/12/2023
20111                                             ;cmp   word [es:di+3Ch], 0 ; [es:di+BDS.opcnt]
20112 00001B69 E8B0FF                             call    chkopcnt
20113 00001B6C 741B                               jz      short checkret ; done if zero
20114
20115                                   ; check for past rom indications. if no rom change indicated, then return ok.
20116
20117                                             ; 26/12/2023
20118                                             ;test   word [es:di+3Fh], 40h
20119                                             ;                       ; test [es:di+BDS.flags], fchanged ; 40h
20120 00001B6E E80E01                             call    checkromchange
20121 00001B71 7416                               jz      short checkret
20122
20123                                   ; we now see that a change line has been seen in the past. let's do the
20124                                   ; expensive verification.
20125
20126 00001B73 E860EB                             call    GetBp           ; build bpb in current bds
20127 00001B76 720F                               jb      short ret_no_error_map ; getbp has already called maperror
20128 00001B78 E81600                             call    check_vid
20129 00001B7B 7207                               jb      short checklatchret ; disk error trying     to read in.
20130 00001B7D 09F6                               or      si, si          ; is changed for sure?
20131 00001B7F 7908                               jns     short checkret
20132 00001B81 E88F00                             call    returnvid
20133                                   checklatchret:
20134 00001B84 E822F2                             call    maperror        ; fix up al for       return to dos
20135                                   ret_no_error_map:
20136 00001B87 F9                                 stc
20137 00001B88 5E                                 pop     si              ; pop off return address
20138                                   checkret:
20139 00001B89 C3                                 retn
20140                                   ; ---------------------------------------------------------------------------
20141
20142                                   ; check the fat and the vid. return in di -1 or 0. return with carry set
20143                                   ; only if there was a disk error. return that error code in ax.
20144                                   ;
20145                                   ;   called with es:di -> bds, ds -> Bios_Data
20146
20147                                   checkfatvid:
20148 00001B8A E8D101                             call    fat_check       ; check the fat and the vid
20149 00001B8D 09F6                               or      si, si
20150 00001B8F 7835                               js      short changed_drv
20151
20152                                   ; the fat was the same. fall into check_vid and check volume id.
20153
20154                                             ; fall into check_vid
20155
20156                                   ; =============== S U B   R O U T I N E =======================================
20157
20158                                   ; now with the extended boot record, the logic should be enhanced.
20159                                   ;
20160                                   ; if it is the extended boot record, then we check the volume serial
20161                                   ; number instead of volume id. if it is different, then set si to -1.
20162                                   ;
20163                                   ; if it is same, then si= 1 (no change).
20164                                   ;
20165                                   ; if it is not the extended boot record, then just follows the old
20166                                   ; logic. dos 4.00 will check if the # of fat in the boot record bpb
20167                                   ; is not 0.  if it is 0 then it must be non_fat based system and
20168                                   ; should have already covered by extended boot structure checking.
20169                                   ; so, we will return "i don't know" by setting si to 0.
20170                                   ;
20171                                   ; this routine assume the newest valid boot record is in cs:[disksector].
20172                                   ; (this will be gauranteed by a successful getbp call right before this
20173                                   ; routine.)
20174                                   ;
20175                                   ;   called with es:di -> bds, ds -> bds
20176
20177                                             ; 26/12/2023 - Retro DOS v5.0
20178                                             ; 19/10/2022
20179                                   check_vid:
20180
20181                                   ; check the disksector.EXT_BOOT_SIG variable for the extended
20182                                   ; boot signature. if it is set then go to do the extended
20183                                   ; id check otherwise continue with code below
20184
20185                                             ; 26/12/2023
20186                                             ;;;
20187 00001B91 833E[6801]00                       cmp     word [disksector+16h], 0 ; BPB_FATSz16
20188 00001B96 7507                               jnz      short chk_vid_1
20189 00001B98 803E[9401]29                       cmp      byte [disksector+42h], 29h ; BS_FAT32_BootSig
20190                                                               ; [disksector+EXT_BOOT.SIG],EXT_BOOT_SIGNATURE
20191 00001B9D EB05                               jmp      short chk_vid_2
20192                                   chk_vid_1:
20193                                             ;;;
20194 00001B9F 803E[7801]29                       cmp     byte [disksector+26h], 29h
20195                                                               ; [disksector+EXT_BOOT.SIG],
20196                                                               ; EXT_BOOT_SIGNATURE
20197                                   chk_vid_2:        ; 26/12/2023
20198 00001BA4 7427                               jz      short do_ext_check_id
20199 00001BA6 E8E200                             call    haschange
20200 00001BA9 74DE                               jz      short checkret
20201 00001BAB 31F6                               xor     si, si
20202 00001BAD 803E[6201]00                       cmp     byte [disksector+10h], 0 ; BPB_NumFATs
20203                                                               ; [disksector+EXT_BOOT.BPB+EBPB.NUMBEROFFATS]
20204 00001BB2 7411                               jz      short checkfatret ; don't read vol id
20205                                                               ; if not fat system
```

```
20206 00001BB4 E8F400                           call    read_volume_id
20207 00001BB7 720C                             jb      short checkfatret
20208 00001BB9 E89901                           call    check_volume_id
20209 00001BBC BEFFFF                           mov     si, 0FFFFh      ; -1
20210                                                                   ; definitely changed
20211 00001BBF 7505                             jnz     short changed_drv
20212
20213 00001BC1 46                               inc     si              ; not changed
20214                                   vid_no_changed:
20215 00001BC2 E8C000                           call    resetchanged
20216                                           ; 12/12/2022
20217                                           ; cf=0 ('and' instruction in 'resetchanged' clears cf)
20218                                           ;clc
20219                                   checkfatret:
20220 00001BC5 C3                               retn
20221                                   ; ---------------------------------------------------------------------------
20222
20223                                           ; 12/12/2022
20224                                   changed_drv:
20225 00001BC6 F8                               clc                     ; cas -- return       no error
20226 00001BC7 C606[1E01]FF                     mov     byte  [tim_drv], 0FFh
20227                                                                   ; ensure that we ask rom for media
20228 00001BCC C3                               retn                    ; check next time round
20229                                   ; ---------------------------------------------------------------------------
20230
20231                                   ; extended id check
20232
20233                                   ; 16/10/2022
20234
20235                                   ; the code to check extended id is basically a check to see if the
20236                                   ; volume serial number is still the same. the volume serial number
20237                                   ; previously read is in cs:disksector.EXT_BOOT_SERIAL
20238                                   ; ds:di points to the bds of the drive under consideration.
20239                                   ; the bds has fields containing the high and low words
20240                                   ; of the volume serial number of the media in the drive.
20241                                   ; compare these fields to the fields mentioned above. if these fields
20242                                   ; do not match the media has changed and so we should jump to the code
20243                                   ; starting at ext_changed else return "i don't know" status
20244                                   ; in the register used for the changeline status and continue executing
20245                                   ; the code given below. for temporary storage use the register which
20246                                   ; has been saved and restored around this block.
20247                                   ;
20248                                   ; bds fields in inc\msbds.inc
20249
20250                                           ; 26/12/2023 - Retro DOS v5.0
20251                                           ; 19/10/2022
20252                                   do_ext_check_id:
20253                                           ; 26/12/2023
20254                                           ;push   ax
20255                                           ;;mov   ax, word ptr ds:disksector+27h
20256                                           ;                       ; [DiskSector+EXT_BOOT.SERIAL]
20257                                           ;mov    ax, [disksector+27h]
20258                                   ; 26/12/2023
20259                                   %if 1
20260                                           ;;;
20261 00001BCD 57                               push    di
20262 00001BCE BE[9501]                         mov     si, disksector+43h ; BS_FAT32_VolID
20263                                                                   ; [DiskSector+FAT32_EXT_BOOT.SERIAL]
20264 00001BD1 833E[6801]00                     cmp     word [disksector+16h], 0 ; BPB_FATSz16
20265 00001BD6 7403                             jz      short chk_vid_3
20266 00001BD8 83EE1C                           sub     si, 28          ; BS_VolID
20267                                           ; si = disksector+27h ; [DiskSector+EXT_BOOT.SERIAL]
20268                                   chk_vid_3:
20269                                           ; [es:di+89h] = [es:di+BDS.vol_serial]
20270 00001BDB 81C78900                         add     di, 137         ; BDS.vol_serial
20271 00001BDF A7                               cmpsw   ; [DiskSector+EXT_BOOT.SERIAL] ; (or FAT32_EXT_BOOT)
20272                                                   ;               = [di+BDS.vol_serial] ?
20273 00001BE0 7501                             jnz     short chk_vid_4
20274 00001BE2 A7                               cmpsw   ; [DiskSector+EXT_BOOT.SERIAL+2] ; (or FAT32_EXT_BOOT)
20275                                                   ;               = [di+BDS.vol_serial+2] ?
20276                                   chk_vid_4:
20277 00001BE3 5F                               pop     di
20278                                           ;pop    ax
20279 00001BE4 7504                             jnz     short ext_changed ; not equal/same
20280 00001BE6 31F6                             xor     si, si          ; 0 ; don't know
20281 00001BE8 EBD8                             jmp     short vid_no_changed ; reset the flag
20282                                           ;;;
20283                                   %else
20284                                           ; 02/09/2023
20285                                           xor     si, si ; 0
20286                                           cmp     ax, [es:di+57h]       ; [di+BDS.vol_serial]
20287                                           jnz     short ext_changed
20288                                           mov     ax, [disksector+29h] ; [DiskSector+EXT_BOOT.SERIAL+2]
20289                                           cmp     ax, [es:di+59h]       ; [di+BDS.vol_serial+2]
20290                                           jnz     short ext_changed
20291                                           ;xor    si, si          ; 0
20292                                                                   ; don't know
20293                                           pop     ax
20294                                           jmp     short vid_no_changed
20295                                                                   ; reset the flag
20296                                   %endif
20297
20298                                   ; ---------------------------------------------------------------------------
20299
20300                                   ext_changed:
20301                                           ; 26/12/2023
20302                                           ;pop    ax
20303                                           ; 02/09/2023
20304                                           ;dec    si ; mov si, 0FFFFh ; -1
20305 00001BEA BEFFFF                           mov     si, 0FFFFh      ; -1
20306                                                                   ; disk changed!
20307                                           ; 12/12/2022
20308                                           ; ('changed_drv' clears cf)
20309                                           ;clc
20310 00001BED EBD7                             jmp     short changed_drv
20311
20312                                   ; ---------------------------------------------------------------------------
20313
20314                                   ; at i/o time, we detected the error. now we need to determine whether the
20315                                   ; media was truly changed or not. we return normally if media change unknown.
20316                                   ; and we pop off the call and jmp to harderr if we see an error.
20317                                   ;
20318                                   ; es:di -> bds
20319
20320                                   checkio:
20321 00001BEF 80FC06                           cmp     ah, 6
20322 00001BF2 75D1                             jnz     short checkfatret
20323 00001BF4 E825FF                           call    chkopcnt
20324 00001BF7 74CC                             jz      short checkfatret
20325 00001BF9 E8DAEA                           call    GetBp
20326 00001BFC 7212                             jb      short no_error_map
20327 00001BFE E889FF                           call    checkfatvid
20328 00001C01 7209                             jb      short checkioret ; disk       error trying to       read in.
20329 00001C03 09F6                             or      si, si          ; is changed for sure?
```

```
20330 00001C05 7802                     js      short checkioerr ; yes changed
20331 00001C07 45                       inc     bp              ; allowa retry
20332 00001C08 C3                       retn
20333                           ; ---------------------------------------------------------------------------
20334
20335                           checkioerr:
20336 00001C09 E80700                   call    returnvid
20337
20338                           checkioret:
20339 00001C0C F9                       stc                     ; make sure carry gets passed through
20340 00001C0D E955F1                   jmp     harderr
20341                           ; ---------------------------------------------------------------------------
20342
20343                           no_error_map:
20344 00001C10 E955F1                   jmp     harderr2
20345
20346                           ; =============== S U B   R O U T I N E =======================================
20347
20348                           ; return vid sets up the vid for a return to dos.
20349                           ;  es:di -> bds, returns pointer in packet to bds_volid
20350                           ;  **** trashes si! ****
20351
20352                           returnvid:
20353 00001C13 BE1600                   mov     si, 22          ; extra
20354                                                           ; offset into pointer to return      value
20355 00001C16 E80700                   call    vid_into_packet
20356 00001C19 B406                     mov     ah, 6
20357 00001C1B F9                       stc
20358 00001C1C C3                       retn
20359
20360                           ; ---------------------------------------------------------------------------
20361
20362                           ; moves the pointer to the volid for the drive into the original request packet
20363                           ; no attempt is made to preserve registers.
20364                           ;
20365                           ; assumes es:di -> bds
20366                           ; **trashes si**
20367
20368                           media_set_vid:
20369 00001C1D BE0F00                   mov     si, 15          ; trans+1
20370                                                           ; return the value here      in packet
20371
20372                                   ; fall into vid_into_packet
20373
20374                           ; =============== S U B   R O U T I N E =======================================
20375
20376                           ; return pointer to vid in bds at es:di in packet[si]
20377
20378                                   ; 14/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
20379                                   ; 19/10/2022
20380                           vid_into_packet:
20381 00001C20 1E                       push    ds              ; return pointer to vid      in bds at es:di       in packet[si]
20382 00001C21 C51E[1200]               lds     bx, [ptrsav]
20383                                   ;add    di, 75          ; BDS.volid
20384                                   ; 14/04/2024
20385 00001C25 83C77D                   add     di, 125 ; (PCDOS 7.1)
20386 00001C28 8938                     mov     [bx+si], di
20387                                   ;sub    di, 75          ; BDS.volid
20388 00001C2A 83EF7D                   sub     di, 125
20389 00001C2D 8C4002                   mov     [bx+si+2], es
20390 00001C30 1F                       pop     ds
20391                           dofloppy:   ; 18/12/2022
20392 00001C31 C3                       retn
20393
20394                           ; ---------------------------------------------------------------------------
20395
20396                           ;---------------------------------------------------------------------------
20397                           ;   hidensity - examine a drive/media descriptor to set the media type. if
20398                           ;   the media descriptor is not f9 (not 96tpi or 3 1/2), we return and let the
20399                           ;   caller do the rest. otherwise, we pop off the return and jump to the tail
20400                           ;   of getbp. for 3.5" media, we just return.
20401                           ;
20402                           ;   inputs: es:di point to correct bds for this drive
20403                           ;           ah has media byte
20404                           ;
20405                           ;   outputs:        carry clear
20406                           ;               no registers modified
20407                           ;           carry set
20408                           ;               al = sectors/fat
20409                           ;               bh = number of root directory entries
20410                           ;               bl = sectors per track
20411                           ;               cx = number of sectors
20412                           ;               dh = sectors per allocation unit
20413                           ;               dl = number of heads
20414                           ;
20415                           ;---------------------------------------------------------------------------
20416
20417                                   ; 26/12/2023 - Retro DOS v5.0
20418                           hidensity:
20419
20420                           ; check for correct drive
20421
20422                                   ; 26/12/2023
20423 00001C32 26F6453F02               test    byte [es:di+3Fh], 2 ; is it special?
20424                                   ; 12/12/2022
20425                                   ;test   byte [es:di+23h], 2
20426                                   ;;test  word [es:di+23h], 2 ; is it special?
20427                                                           ; [es:di+BDS.flags], fchangeline
20428 00001C37 74F8                     jz      short dofloppy ; no, do normal floppy test
20429
20430                           ; we have a media byte that is pretty complex. examine drive information
20431                           ; table to see what kind it is.
20432
20433                                   ; 26/12/2023
20434 00001C39 26807D3E02               cmp     byte [es:di+3Eh], 2 ; is it single-media?
20435                                   ;cmp    byte [es:di+22h], 2 ; is it single-media?
20436 00001C3E 74F1                     jz      short dofloppy ; [es:di+BDS.formfactor], ffSmall
20437                                                           ; yes, use fatid...
20438                           ; 96 tpi drive?
20439 00001C40 80FCF9                   cmp     ah, 0F9h
20440 00001C43 75EC                     jnz     short dofloppy
20441
20442                           ;------ If formfactor of drive = ffother or ff288 it has to be
20443                           ;------ a 720K diskette
20444
20445                                   ; 02/09/2023 (PCDOS 7.1)
20446                                   ; 26/12/2023
20447 00001C45 268A453E                 mov     al, [es:di+3Eh] ; [es:di+BDS.formfactor]
20448                                   ;mov    al, [es:di+22h]       ; [es:di+BDS.formfactor]
20449 00001C49 3C07                     cmp     al, 7
20450                                   ;cmp    byte [es:di+22h], 7 ; [es:di+BDS.formfactor]
20451                                                           ; ffOther
20452 00001C4B 7413                     jz      short Is720K
20453 00001C4D 3C09                     cmp     al, 9
```

```
20454                                           ;cmp    byte [es:di+22h], 9 ; [es:di+BDS.formfactor]
20455                                                                   ; ff288
20456 00001C4F 740F                             jz      short Is720K
20457 00001C51 B007                             mov     al, 7           ; seven sectors / fat
20458 00001C53 BB0FE0                           mov     bx, 57359       ; 224*256+0Fh
20459                                                                   ; 224 root dir entries
20460                                                                   ; & 0Fh sector max
20461 00001C56 B96009                           mov     cx, 2400        ; 80*15*2
20462                                                                   ; 80 tracks, 15 sectors/track,
20463                                                                   ; 2 sides
20464                                           ; 02/09/2023
20465 00001C59 5A                               pop     dx              ; pop off return address
20466 00001C5A BA0201                           mov     dx, 258         ; 1*256+2
20467                                                                   ; sectors/allocation unit
20468                                                                   ; & head max
20469                                           ;add    sp, 2           ; pop off return address
20470 00001C5D E9EAEA                           jmp     Has1            ; return to tail of getbp
20471                                   ; ---------------------------------------------------------------------------
20472
20473                                   Is720K:
20474                                           ; 02/09/2023
20475 00001C60 5B                               pop     bx              ; pop off return address
20476                                           ;add    sp, 2           ; pop off return address
20477 00001C61 E9A9EA                           jmp     Has720K         ; return to 720K code
20478                                   ; ---------------------------------------------------------------------------
20479
20480                                           ; 18/12/2022
20481                                   ;dofloppy:
20482                                           ;retn
20483
20484                                   ; =============== S U B   R O U T I N E =======================================
20485
20486                                   ; 16/10/2022
20487
20488                                   ;---------------------------------------------------------------------------
20489                                   ; set_changed_dl - sets flag bits according to bits set in bx.
20490                                   ;             essentially used to indicate changeline, or format.
20491                                   ;
20492                                   ;   inputs: dl contains physical drive number
20493                                   ;           bx contains bits to set in the flag field in the bdss
20494                                   ;   outputs:       none
20495                                   ;   registers modified: flags
20496                                   ;
20497                                   ;   called from int13 hooker.  Must preserve ALL registers!!!
20498                                   ;
20499                                   ; in the virtual drive system we *must* flag the other drives as being changed
20500                                   ;---------------------------------------------------------------------------
20501
20502                                           ; 26/12/2023 - Retro DOS v5.0
20503                                   set_changed_dl:
20504 00001C64 06                               push    es
20505 00001C65 57                               push    di
20506                                           ;les    di, ds:start_bds
20507                                           ; 19/10/2022
20508 00001C66 C43E[1901]                       les     di, [start_bds]
20509
20510                                   ; note: we assume that the list is non-empty
20511
20512                                   scan_bds:
20513 00001C6A 26385504                         cmp     [es:di+4], dl  ; [es:di+BDS.drivenum]
20514 00001C6E 7504                             jnz     short get_next_bds
20515
20516                                   ; someone may complain, but this *always* must be done when a disk change is
20517                                   ; noted. there are *no* other compromising circumstances.
20518
20519                                           ; 26/12/2023
20520 00001C70 26095D3F                         or      [es:di+3Fh], bx       ; [es:di+BDS.flags]
20521                                           ;or     [es:di+23h], bx       ; [es:di+BDS.flags]
20522                                                                   ; signal change       on other drive
20523                                   get_next_bds:
20524 00001C74 26C43D                           les     di, [es:di]     ; [es:di+BDS.link]
20525                                                                   ; go to next bds
20526 00001C77 83FFFF                           cmp     di, 0FFFFh
20527 00001C7A 75EE                             jnz     short scan_bds ; loop unless we hit end of chain
20528 00001C7C 5F                               pop     di
20529 00001C7D 07                               pop     es
20530 00001C7E C3                               retn
20531
20532                                   ; =============== S U B   R O U T I N E =======================================
20533
20534                                   ;---------------------------------------------------------------------------
20535                                   ; checkromchange - see if external program has diddled rom change line.
20536                                   ;
20537                                   ;   inputs: es:di points to current bds.
20538                                   ;   outputs:       zero set - no change
20539                                   ;           zero reset - change
20540                                   ;   registers modified: none
20541                                   ;---------------------------------------------------------------------------
20542
20543                                           ; 26/12/2023 - Retro DOS v5.0
20544                                   checkromchange:
20545                                           ;test   word [es:di+BDS.flags], fchanged ; 40h
20546                                           ; 26/12/2023
20547 00001C7F 26F6453F40                       test    byte [es:di+3Fh], 40h
20548                                           ; 10/12/2022
20549                                           ;test   byte [es:di+23h], 40h
20550                                           ;;test  word [es:di+23h], 40h ; [es:di+BDS.flags]
20551                                                                   ; fchanged
20552 00001C84 C3                               retn
20553
20554                                   ; =============== S U B   R O U T I N E =======================================
20555
20556                                   ;---------------------------------------------------------------------------
20557                                   ; resetchanged - restore value of change line
20558                                   ;
20559                                   ;   inputs: es:di points to current bds
20560                                   ;   outputs:       none
20561                                   ;   registers modified: none
20562                                   ;---------------------------------------------------------------------------
20563
20564                                           ; 26/12/2023 - Retro DOS v5.0
20565                                   resetchanged:
20566                                           ;and    word [es:di+BDS.flags], ~fchanged ; 0FFBFh
20567                                           ; 26/12/2023
20568 00001C85 2680653FBF                       and     byte [es:di+3Fh], 0BFh
20569                                           ; 10/12/2022
20570                                           ;and    byte [es:di+23h], 0BFh
20571                                           ;;and   word [es:di+23h], 0FFBFh ; [es:di+BDS.flags]
20572                                                                   ; ~fchanged
20573 00001C8A C3                               retn
20574
20575                                   ; =============== S U B   R O U T I N E =======================================
20576
20577                                   ;---------------------------------------------------------------------------
```

```
                                ; haschange - see if drive can supply change line
                                ;
                                ;   inputs: es:di points to current bds
                                ;   outputs:       zero set - no change line available
                                ;          zero reset - change line available
                                ;   registers modified: none
                                ;-----------------------------------------------------------------------------

                                           ; 26/12/2023 - Retro DOS v5.0
                                haschange:
                                           ;test  word [es:di+BDS.flags], fchangeline ; 2
                                           ; 26/12/2023
00001C8B 26F6453F02                        test   byte [es:di+3Fh], 2
                                           ; 10/12/2022
                                           ;test  byte [es:di+23h], 2
                                           ;;test  word [es:di+23h], 2 ; [es:di+BDS.flags]
                                                                       ; fchangeline
00001C90 C3                                retn

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

                                ; 16/10/2022

                                ;-----------------------------------------------------------------------
                                ; set_volume_id      -    main routine, calls other routines.
                                ; read_volume_id     -    read the volume id and tells if it has been changed.
                                ; transfer_volume_id -    copy the volume id from tmp to special drive.
                                ; check_volume_id    -    compare volume id in tmp area with one expected for drive.
                                ; fat_check          -    see of the fatid has changed in the specified drive.
                                ;-----------------------------------------------------------------------

                                ; set_volume_id
                                ;   if drive has changeline support, read in and set the volume_id
                                ; and the last fat_id byte. if no change line support then do nothing.
                                ;
                                ;   on entry:
                                ;   es:di points to the bds for this disk.
                                ;   ah contains media byte
                                ;
                                ;   on exit:
                                ;   carry clear:
                                ;      successful call
                                ;   carry set
                                ;      error and ax has error code

                                set_volume_id:
00001C91 52                                push   dx             ; save registers
00001C92 50                                push   ax
00001C93 E8F5FF                            call   haschange      ; does drive have changeline support?
00001C96 740B                              jz     short setvret  ; no, get out
00001C98 E81000                            call   read_volume_id
00001C9B 7209                              jb     short seterr
00001C9D E8A900                            call   transfer_volume_id ; copy the volume id     to special drive
00001CA0 E8E2FF                            call   resetchanged   ; restore value       of change line
                                setvret:
                                           ; 10/12/2022
                                           ; cf = 0
                                           ;clc                  ; no error, clear carry flag
00001CA3 58                                pop    ax             ; restore registers
00001CA4 5A                                pop    dx
00001CA5 C3                                retn
                                ; ---------------------------------------------------------------------------

                                seterr:
00001CA6 5A                                pop    dx             ; pop stack but don't overwrite ax
00001CA7 5A                                pop    dx             ; restore dx
00001CA8 C3                                retn
                                ; ---------------------------------------------------------------------------
00001CA9 0000                   root_sec:  dw 0                   ; root sector #

                                ; 16/10/2022
                                ;ROOTSEC equ root_sec - DOSBIOSEG_2C7h
                                ; 09/12/2022
                                ROOTSEC equ root_sec

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

                                ; 16/10/2022

                                ; read_volume_id read the volume id and tells if it has been changed.
                                ;
                                ;   on entry:
                                ;   es:di points to current bds for drive.
                                ;
                                ;   on exit:
                                ;   carry clear
                                ;       si = 1  no change
                                ;       si = 0  ?
                                ;       si = -1 change
                                ;
                                ;   carry set:
                                ;       error and ax has error code.

                                read_volume_id:
00001CAB 52                                push   dx             ; preserve registers
00001CAC 51                                push   cx
00001CAD 53                                push   bx
00001CAE 50                                push   ax
00001CAF 06                                push   es             ; stack the bds last
00001CB0 57                                push   di
00001CB1 1E                                push   ds             ; point es to Bios_Data
00001CB2 07                                pop    es
00001CB3 BF[4008]                          mov    di, tmp_vid    ; "NO NAME      "
00001CB6 BE[6305]                          mov    si, nul_vid    ; "NO NAME      "
                                           ; 26/12/2023
00001CB9 B90B00                            mov    cx, 11         ; PCDOS 7.1 - 02/09/2023
                                           ;mov   cx, 12         ; initialize tmp_vid to      null vi_id

                                           ;rep   movsb
                                           ; 26/12/2023
                                           ;rep movs byte ptr es:[di], byte ptr cs:[si]
                                           ;db 0FBh,2Eh,0A4h
                                           ;cs    ; nul_vid is in BIOSCODE segment
                                           ;rep movsb
00001CBC F3                                rep
00001CBD 2E                                cs
00001CBE A4                                movsb

00001CBF 5F                                pop    di
00001CC0 07                                pop    es
00001CC1 268A450B                          mov    al, [es:di+11] ; [es:di+BDS.fats]
                                                                  ; # of fats
00001CC5 268B4D11                          mov    cx, [es:di+17] ; [es:di+BDS.fatsecs]
                                                                  ; sectors / fat
```

```
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.?)
```

```
20826                                           ;add   di, 75         ; BDS.volid
20827                                           ;mov   cx, 12         ; VOLID_SIZ
20828                                           ;
20829 00001D47 FC                               cld
20830 00001D48 C3                               retn
20831
20832                                   ; ============== S U B   R O U TI N E =======================================
20833
20834                                   ; transfer_volume_id - copy the volume id from tmp to special drive
20835                                   ;
20836                                   ; inputs:   es:di has current bds
20837                                   ; outputs:  bds for drive has volume id from tmp
20838
20839                                           ; 27/12/2023 - Retro DOS v5.0
20840                                   transfer_volume_id:
20841 00001D49 57                               push   di             ; copy the volume id from tmp to special drive
20842                                           ;push  si
20843 00001D4A 51                               push   cx
20844                                           ; 27/12/2023
20845 00001D4B 56                               push   si
20846
20847                                           ;mov   si, tmp_vid    ; "NO NAME      "
20848                                           ;;add  di, BDS.volid
20849                                           ;add   di, 75         ; BDS.volid
20850                                           ;;mov  cx, VOLID_SIZ
20851                                           ;mov   cx, 12         ; VOLID_SIZ
20852                                           ;cld
20853                                           ; 02/09/2023 (PCDOS 7.1)
20854 00001D4C E8EFFF                           call   preset_volid_addr
20855
20856 00001D4F F3A4                             rep movsb
20857
20858                                           ; 27/12/2023
20859 00001D51 5E                               pop    si
20860                                   chk_volid_ok:
20861 00001D52 59                               pop    cx
20862                                           ;pop   si
20863 00001D53 5F                               pop    di
20864 00001D54 C3                               retn
20865
20866                                   ; ============== S U B   R O U TI N E =======================================
20867
20868                                   ;  check_volume_id - compare volume id in tmp area with
20869                                   ;               one expected for drive
20870                                   ;
20871                                   ;   inputs: es:di has current bds for drive
20872                                   ;   outputs:       zero true means it matched
20873
20874                                           ; 27/12/2023 - Retro DOS v5.0
20875                                   check_volume_id:
20876 00001D55 57                               push   di
20877 00001D56 51                               push   cx
20878
20879                                           ;mov   si, tmp_vid    ; "NO NAME      "
20880                                           ;;add  di, BDS.volid
20881                                           ;add   di, 75         ; BDS.volid
20882                                           ;;mov  cx, VOLID_SIZ
20883                                           ;mov   cx, 12         ; VOLID_SIZ
20884                                           ;cld
20885                                           ; 02/09/2023 (PCDOS 7.1)
20886 00001D57 E8E4FF                           call   preset_volid_addr
20887
20888 00001D5A F3A6                             repe cmpsb            ; are the 2 volume_ids the same?
20889
20890                                           ; 27/12/2023
20891                                           ;pop   cx
20892                                           ;pop   di
20893                                           ;retn
20894 00001D5C EBF4                             jmp    short chk_volid_ok
20895
20896                                   ; ============== S U B   R O U TI N E =======================================
20897
20898                                   ;   fat_check - see of the fatid has changed in the specified drive.
20899                                   ;         - uses the fat id obtained from the boot sector.
20900                                   ;
20901                                   ;   inputs: medbyt is expected fat id
20902                                   ;           es:di points to current bds
20903                                   ;
20904                                   ;   output: si = -1 if fat id different,
20905                                   ;           si = 0 otherwise
20906                                   ;
20907                                   ;   no other registers changed.
20908
20909                                   fat_check:
20910 00001D5E 50                               push   ax
20911 00001D5F 31F6                             xor    si, si         ; say fat id's are same.
20912 00001D61 A0[1F01]                         mov    al, [medbyt]   ; 19/10/2022
20913 00001D64 263A4510                         cmp    al, [es:di+10h]       ; [es:di+BDS.media]
20914                                                                 ; compare it with the bds medbyte
20915 00001D68 7401                             jz     short okret1   ; carry clear
20916 00001D6A 4E                               dec    si
20917                                   okret1:
20918 00001D6B 58                               pop    ax
20919 00001D6C C3                               retn
20920
20921                                   ; ---------------------------------------------------------------------------
20922
20923                                   ; BIOSCODE:1DFEh (PCDOS 7.1 IBMBIO.COM) ; 27/12/2023
20924                                           ;times 2 db 0
20925
20926                                   ; BIOSCODE:1A69h (MSDOS 6.21 IO.SYS) ((& MSDOS 6.22 IO.SYS))
20927                                           ;times 7 db 0
20928
20929                                   ; BIOSCODE:180Bh (MSDOS 5.0 IO.SYS)
20930
20931                                           ; 09/12/2022
20932                                           ;times 4 db 0  ; 17/10/2022
20933                                           ;db 4 dup(0)   ; times 4 db 0
20934
20935                                   ; ---------------------------------------------------------------------------
20936
20937                                           ; 09/12/2022
20938                                           ;db 0
20939
20940                                   number2div  equ ($-BCode_start)
20941                                   number2mod  equ (number2div % 16)
20942
20943                                   %if number2mod>0 & number2mod<16
20944 00001D6D 00<rep 3h>                       times (16-number2mod) db 0
20945                                   %endif
20946
20947                                   ;align 16
20948
20949                                   ; 09/12/2022
```

```
20950                                   BCODE_END   equ $ - BCode_start
20951                                   ; 31/03/2024
20952                                   BCODEEND:
20953                                   ;SYSINITSEG equ IOSYSCODESEG+(BCODE_END>>4)
20954                                   ; 13/12/2022
20955                                   SYSINITOFFSET      equ BCODE_END
20956                                   SYSINITSEG  equ IOSYSCODESEG+(SYSINITOFFSET>>4)
20957                                   
20958                                   ; 30/12/2022 - Retro DOS v4.2
20959                                   ; (SYSINITSEG is 473h for MSDOS 6.21 IO.SYS)
20960                                   
20961                                   ;--- End of DOSBIOS code segment ---------------------------------------------
20962                                   
20963                                   ; 16/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
20964                                   ; 01/05/2019 - Retro DOS v4.0
20965                                   ; ============================================================================
20966                                   ; end of BIOSCODE
20967                                   
20968                                   ; ----------------------------------------------------------------------------
20969                                   ; %include sysinit5.s     ; 09/12/2022
20970                                   ; ----------------------------------------------------------------------------
20971                                   
20972                                   ;============================================================================
20973                                   ; (IO.SYS) SYSINIT SEGMENT
20974                                   ;============================================================================
20975                                   ; 09/12/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
20976                                   ; 30/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
20977                                   ;
20978                                   ; 27/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
20979                                   
20980                                   section .SYSINIT vstart=0
20981                                   
20982                                   ; ****************************************************************************
20983                                   ; SYSINIT.BIN (MSDOS 6.21 IO.SYS) - RETRO DOS v4.0 by ERDOGAN TAN - 21/10/2022
20984                                   ; ----------------------------------------------------------------------------
20985                                   ; Last Update: 04/01/2023 (Modified IO.SYS)  ((Previous: 30/12/2022))
20986                                   ; ----------------------------------------------------------------------------
20987                                   ; Beginning: 03/06/2018 (Retro DOS 3.0), 21/03/2019 (Retro DOS 4.0)
20988                                   ; ----------------------------------------------------------------------------
20989                                   ; Assembler: NASM version 2.15
20990                                   ; ----------------------------------------------------------------------------
20991                                   ;      ((nasm sysinit6.s -l sysinit6.lst -o SYSINIT6.BIN -Z error.txt))
20992                                   ; ----------------------------------------------------------------------------
20993                                   ; Modified from 'sysinit2.s' (SYSINIT2.BIN) file of Retro DOS v3.0 (6/7/2018)
20994                                   ; ----------------------------------------------------------------------------
20995                                   ; Derived from 'SYSINIT1.ASM' and 'SYSINIT2.ASM' files of MSDOS 6.0
20996                                   ; source code by Microsoft, 1991
20997                                   ; ----------------------------------------------------------------------------
20998                                   ; Derived from 'SYSINIT.ASM' file of MSDOS 2.0 (IBM PCDOS v2.0) source code
20999                                   ; by Microsoft, 12/10/1983
21000                                   ; ****************************************************************************
21001                                   ; main file: 'retrodos4.s'
21002                                   ; incbin 'SYSINIT3.BIN' ; (SYINITSEG)
21003                                   
21004                                   ; 30/12/2022 - Retro DOS v4.2
21005                                   ; Retro DOS v4.0 - 2019
21006                                   ; SYSINIT (MSDOS 6.21 IO.SYS) draft: 'sysinit3.s' (01/07/2019)
21007                                   
21008                                   ; 21/10/2022
21009                                   ; ----------------------------------------------------------------------------
21010                                   ; This source code (version) is based on SYSINIT source code of disassembled
21011                                   ; MSDOS 5.0 IO.SYS file (SYSINIT.BIN)
21012                                   ; Dissassembler: Hex-Rays Interactive Disassembler (IDA)
21013                                   ; ----------------------------------------------------------------------------
21014                                   ; Binary file splitter & joiner: FFSJ v3.3
21015                                   
21016                                   ;--------------------------------------------------------------
21017                                   ; SYSINIT.TXT (27/01/1983)
21018                                   ;--------------------------------------------------------------
21019                                   ;    SYSINIT is  a module linked behind the OEM bios.  It takes
21020                                   ;over  the  system  initialization  after  the  OEM  bios   has
21021                                   ;performed any  initialization  it  needs  to  do.   Control is
21022                                   ;transfered with a long jump to the external  variable  SYSINIT
21023                                   ;
21024                                   ;
21025                                   ;   The OEM  has  the  following  variables declared external:
21026                                   ;
21027                                   ;   CURRENT_DOS_LOCATION    WORD
21028                                   ;
21029                                   ;This word  contains  the  segment  number of the DOS before it
21030                                   ;is relocated.  The OEM bios must set this value.
21031                                   ;
21032                                   ;   FINAL_DOS_LOCATION      WORD
21033                                   ;
21034                                   ;This word contains the segment number of the DOS after SYSINIT
21035                                   ;moves it.  The OEM bios must set this value.
21036                                   ;
21037                                   ;   DEVICE_LIST             DWORD
21038                                   ;
21039                                   ;This  double  word  pointer  points  to  the  linked  list  of
21040                                   ;character and block device drivers.  The  OEM  must  set  this
21041                                   ;value.
21042                                   ;
21043                                   ;   MEMORY_SIZE             WORD
21044                                   ;
21045                                   ;This word  contains  the  number  of  RAM  paragraphs.  If the
21046                                   ;bios doesn't set  this  variable  SYSINIT  will  automatically
21047                                   ;calculate it.   NOTE:  systems with PARITY checked memory must
21048                                   ;size memory in the BIOS.  SYSINITs method is to  write  memory
21049                                   ;and read it back until it gets a mismatch.
21050                                   ;
21051                                   ;   DEFAULT_DRIVE           BYTE
21052                                   ;
21053                                   ;This is  the initial default drive when the system first comes
21054                                   ;up.  drive a=0, drive b=1,  etc.   If  the  bios  doesn't  set
21055                                   ;it then drive a is assumed.
21056                                   ;
21057                                   ;   BUFFERS                 BYTE
21058                                   ;
21059                                   ;This is  the  default  number of buffers for the system.  This
21060                                   ;value may be overridden by the user in  the  CONFIG.SYS  file.
21061                                   ;It is DBed to 2 in SYSINIT it should be greater than 1.
21062                                   ;
21063                                   ;   FILES                   BYTE
21064                                   ;
21065                                   ;This is  the  default  number  of  files for the system.  This
21066                                   ;value may be overridden by the user in  the  CONFIG.SYS  file.
21067                                   ;It is  DBed  to  8 in SYSINIT, values less than 5 are ignored.
21068                                   ;
21069                                   ;   SYSINIT                 FAR
21070                                   ;
21071                                   ;The entry  point  of  the  SYSINIT  module.  OEM BIOS jumps to
21072                                   ;this label at the end of its INIT code.
21073                                   ;
```

```
21074                                   ;   The OEM  has  the  following  variables declared public:
21075                                   ;
21076                                   ;   RE_INIT                FAR
21077                                   ;
21078                                   ;This is an entry point which allows the BIOS to do some INIT
21079                                   ;work  after  the  DOS is initialized.  ALL REGISTERS MUST BE
21080                                   ;PRESERVED.  On entry DS points to the first available memory
21081                                   ;(after  the DOS).  DS:0 points to a 100H byte program header
21082                                   ;prefix which represents  the  "program"  currently  running.
21083                                   ;This  program  should  be  thought  of  as  the OEM BIOS and
21084                                   ;SYSINIT taken together.  This is not  a  normal  program  in
21085                                   ;that  no  memory  is  allocated to it, it is running in free
21086                                   ;memory.
21087                                   ;NOTES:
21088                                   ;     At the time this routine is called SYSINIT occupies the
21089                                   ;highest 10K of memory ("highest" is determined by the  value
21090                                   ;of the MEMORY_SIZE variable), DO NOT DO WRITES THERE.
21091                                   ;     Since this is called AFTER DOS is initialized, you can
21092                                   ;make system calls.  This also implies that the code for this
21093                                   ;routine    CANNOT   be   thrown   away   by   use   of   the
21094                                   ;FINAL_DOS_LOCATION since the DOS has already been moved.
21095                                   ;     If you don't want  anything done just set this to point
21096                                   ;at a FAR RET instruction.
21097
21098                                   ; --------------------------------------------------------------------------
21099                                   ; TITLE   BIOS SYSTEM INITIALIZATION
21100                                   ; --------------------------------------------------------------------------
21101
21102                                   ;include version.inc
21103                                   ; --------------------------------------------------------------------------
21104
21105                                   ;FALSE   EQU     0
21106                                   ;TRUE    EQU     0FFFFh
21107
21108                                   ;IBMVER         EQU     TRUE
21109                                   ;IBMCOPYRIGHT EQU   FALSE
21110                                   ;STACKSW    EQU        TRUE              ;Include Switchable Hardware Stacks
21111                                   ;IBMJAPVER  EQU     FALSE         ; If TRUE set KANJI true also
21112                                   ;MSVER      EQU     FALSE
21113                                   ;ALTVECT    EQU     FALSE         ; Switch to build ALTVECT version
21114                                   ;KANJI      EQU     FALSE
21115
21116                                   ;(MSDOS 6.0, versiona.inc, 1991)
21117                                   ; --------------------------------------------------------------------------
21118                                   ;MAJOR_VERSION  EQU       6
21119                                   ;;MINOR_VERSION    EQU     0       ;6.00
21120                                   ;MINOR_VERSION  EQU       21      ;6.21  ; 21/03/2019 - Retro DOS v4.0
21121
21122                                   ; 22/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0)
21123                                   ; --------------------------------------------------------------------------
21124                                   ;MAJOR_VERSION   EQU      5
21125                                   ;MINOR_VERSION   EQU      0
21126
21127                                   ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21)
21128                                   ;MAJOR_VERSION   EQU      6
21129                                   ;MINOR_VERSION   EQU      22
21130
21131                                   ; 21/02/2024 - Retro DOS v5.0 (Modified PCDOS 7.1)
21132                                   MAJOR_VERSION   EQU       7
21133                                   MINOR_VERSION   EQU       10
21134
21135                                   expected_version equ      (MINOR_VERSION<<8)+MAJOR_VERSION
21136
21137                                   ;DOSREVNM equ      00000000b       ; m037 - bits 0-2 = revision number of DOS
21138                                                      ; currently 0.
21139                                   DOSREVNM equ       00000111b      ; [[[ 7 for Retro DOS v4.0 ]]] (21/03/2019)
21140                                   DOSINROM equ       00001000B       ; bit 3 of ver flags returned in BH
21141                                   DOSINHMA equ       00010000B       ; bit 4 of ver flags
21142
21143                                   ;      if1
21144                                   ;      %OUT  ... for DOS Version 5.00 ...
21145                                   ;      endif
21146
21147                                          ;*****************************
21148                                          ;Each assembler program should:
21149                                          ;  mov ah,030h                  ;DOS Get Version function
21150                                          ;  int 021h                    ;Version ret. in AX,minor version first
21151                                          ;  cmp ax,expected_version      ;ALL utilities should check for an
21152                                          ;  jne error_handler           ; EXACT version match.
21153                                          ;*****************************
21154
21155                                   ; --------------------------------------------------------------------------
21156                                   ; device definitions
21157
21158                                   ;Attribute bit masks
21159                                   DEVTYP  EQU     8000h           ;Bit 15 - 1  if Char, 0 if block
21160                                   DEVIOCTL EQU    4000h           ;Bit 14 - CONTROL mode bit
21161                                   ISFATBYDEV EQU  2000h           ;Bit 13 - Device uses FAT ID bytes, comp media.
21162                                   ISCIN   EQU     0001h           ;Bit 0 - This device is the console input.
21163                                   ISCOUT  EQU     0002h           ;Bit 1 - This device is the console output.
21164                                   ISNULL  EQU     0004h           ;Bit 2 - This device is the null device.
21165                                   ISCLOCK EQU     0008h           ;Bit 3 - This device is the clock device.
21166                                   ISIBM   EQU     0010h           ;Bit 4 - This device is special
21167
21168                                   ; The device table list has the form:
21169                                   struc       SYSDEV
21170 00000000 ????????                .NEXT:              resd 1          ;Pointer to next device header
21171 00000004 ????                    .ATT:               resw 1          ;Attributes of the device
21172 00000006 ????                    .STRAT:             resw 1          ;Strategy entry point
21173 00000008 ????                    .INT:               resw 1          ;Interrupt entry point
21174 0000000A ????????????????        .NAME:              resb 8          ;Name of device (only first byte used for block)
21175                                   .size:
21176                                   endstruc
21177
21178                                   ;Static Reguest Header
21179                                   struc       SRHEAD
21180 00000000 ??                      .REQLEN:    resb 1          ;Length in bytes of request block
21181 00000001 ??                      .REQUNIT:   resb 1          ;Device unit number
21182 00000002 ??                      .REQFUNC:   resb 1          ;Type of request
21183 00000003 ????                    .REQSTAT:   resw 1          ;Status Word
21184 00000005 ????????????????                   resb 8          ;Reserved for queue links
21185                                   .size:
21186                                   endstruc
21187
21188                                   ;Status word masks
21189                                   STERR   EQU     8000H           ;Bit 15 - Error
21190                                   STBUI   EQU     0200H           ;Bit 9 - Buisy
21191                                   STDON   EQU     0100H           ;Bit 8 - Done
21192                                   STECODE EQU     00FFH           ;Error code
21193                                   WRECODE EQU     0
21194
21195                                   ;Function codes
21196                                   DEVINIT EQU     0               ;Initialization
21197                                   DINITHL EQU     26              ;Size of init header
```

```
21198                                   DEVMDCH EQU     1               ;Media check
21199                                   DMEDHL  EQU     15              ;Size of media check header
21200                                   DEVBPB  EQU     2               ;Get BPB
21201                                   DEVRDIOCTL EQU  3               ;IOCTL read
21202                                   DBPBHL  EQU     22              ;Size of Get BPB header
21203                                   DEVRD   EQU     4               ;Read
21204                                   DRDWRHL EQU     22              ;Size of RD/WR header
21205                                   DEVRDND EQU     5               ;Non destructive read no wait (character devs)
21206                                   DRDNDHL EQU     14              ;Size of non destructive read header
21207                                   DEVIST  EQU     6               ;Input status
21208                                   DSTATHL EQU     13              ;Size of status header
21209                                   DEVIFL  EQU     7               ;Input flush
21210                                   ; 21/02/2024
21211                                   ;DFLSHL EQU     15              ;Size of flush header
21212                                   DFLSHL      equ    13     ; PCDOS 7.1 IBMDOS.COM  ; 21/02/2024
21213                                   DEVWRT  EQU     8               ;Write
21214                                   DEVWRTV EQU     9               ;Write with verify
21215                                   DEVOST  EQU     10              ;Output status
21216                                   DEVOFL  EQU     11              ;Output flush
21217                                   DEVWRIOCTL EQU  12              ;IOCTL write
21218
21219                                   ; ----------------------------------------------------------------------------
21220                                   struc       SYS_FCB
21221 00000000 ??                      .fcb_drive: resb 1
21222 00000001 ????????????????        .fcb_name:  resb 8
21223 00000009 ??????                  .fcb_ext:   resb 3
21224 0000000C ????                    .fcb_EXTENT:       resw 1
21225 0000000E ????                    .fcb_RECSIZ:       resw 1  ; Size of record (user settable)
21226 00000010 ????                    .fcb_FILSIZ:       resw 1  ; Size of file in bytes; used with the following
21227                                                              ; word
21228 00000012 ????                    .fcb_DRVBP: resw 1  ; BP for SEARCH FIRST and SEARCH NEXT
21229 00000014 ????                    .fcb_FDATE: resw 1  ; Date of last writing
21230 00000016 ????                    .fcb_FTIME: resw 1  ; Time of last writing
21231 00000018 ??                      .fcb_DEVID: resb 1  ; Device ID number, bits 0-5 if file.
21232                                                              ; bit 7=0 for file, bit 7=1 for I/O device
21233                                                              ; If file, bit 6=0 if dirty
21234                                                              ; If I/O device, bit 6=0 if EOF (input)
21235                                                              ;               Bit 5=1 if Raw mode
21236                                                              ;               Bit 0=1 if console input device
21237                                                              ;               Bit 1=1 if console output device
21238                                                              ;               Bit 2=1 if null device
21239                                                              ;               Bit 3=1 if clock device
21240 00000019 ????                  .fcb_FIRCLUS:      resw 1  ; First cluster of file
21241 0000001B ????                  .fcb_CLUSPOS:      resw 1  ; Position of last cluster accessed
21242 0000001D ????                  .fcb_LSTCLUS:      resw 1  ; Last cluster accessed and directory
21243 0000001F ??                                       resb 1  ; pack 2 12 bit numbers into 24 bits...
21244 00000020 ??                    .fcb_NR:    resb 1  ; Next record
21245 00000021 ????????              .fcb_RR:    resb 4  ; Random record
21246                                 .size:
21247                                 endstruc
21248
21249                                 ; ----------------------------------------------------------------------------
21250                                 ; Field definition for I/O buffer information
21251
21252                                 ; 22/03/2019 - Retro DOS v4.0 (MSDOS 6.0, BUFFER.INC, 1991)
21253
21254                                     ; 03/01/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMDOS.COM)
21255
21256                                 struc BUFFINFO
21257 00000000 ????                  .buf_next:  resw 1          ; Pointer to next buffer in list
21258 00000002 ????                  .buf_prev:  resw 1          ; Pointer to prev buffer in list
21259 00000004 ??                    .buf_ID:    resb 1          ; Drive of buffer (bit 7 = 0)
21260                                                            ; SFT table index (bit 7 = 1)
21261                                                            ; = FFH if buffer free
21262 00000005 ??                    .buf_flags: resb 1          ; Bit 7 = 1 if Remote file buffer
21263                                                            ;       = 0 if Local device buffer
21264                                                            ; Bit 6 = 1 if buffer dirty
21265                                                            ; Bit 5 = Reserved
21266                                                            ; Bit 4 = Search bit (bit 7 = 1)
21267                                                            ; Bit 3 = 1 if buffer is DATA
21268                                                            ; Bit 2 = 1 if buffer is DIR
21269                                                            ; Bit 1 = 1 if buffer is FAT
21270                                                            ; Bit 0 = Reserved
21271 00000006 ????????              .buf_sector:       resd 1          ; Sector number of buffer (flags bit 7 = 0)
21272                                 ; The next two items are often refed as a word (flags bit 7 = 0)
21273 0000000A ??                    .buf_wrtcnt:       resb 1          ; For FAT sectors, # times sector written out
21274 0000000B ????                  .buf_wrtcntinc:    resw 1          ; "   "     "   , # sectors between each write
21275 0000000D ????                              resw 1 ; * ; 03/01/2024 ; PCDOS 7.1
21276                                                       ; hw of sectors per FAT
21277 0000000F ????????              .buf_DPB:   resd 1          ; Pointer to drive parameters
21278 00000013 ????                  .buf_fill:  resw 1          ; How full buffer is (flags bit 7 = 1)
21279 00000015 ??                    .buf_reserved:     resb 1          ; make DWORD boundary for 386
21280 00000016 ????                              resw 1 ; * ; 03/01/2024 ; PCDOS 7.1
21281                                                       ; reserved word for dword boundary
21282                                 .size:      ; 20 bytes ; MSDOS 5.0 to 6.22
21283                                     ; 24 bytes ; PCDOS 7.1 ; 03/01/2024
21284                                 endstruc
21285
21286                                 %define buf_offset BUFFINFO.buf_sector ; 22/07/2019
21287                                                    ;For buf_flags bit 7 = 1, this is the byte
21288                                                    ;offset of the start of the buffer in
21289                                                    ;the file pointed to by buf_ID. Thus
21290                                                    ;the buffer starts at location
21291                                                    ;buf_offset in the file and contains
21292                                                    ;buf_fill bytes.
21293
21294                                 bufinsiz    equ     BUFFINFO.size  ; Size of structure in bytes
21295
21296
21297                                 buf_Free    equ     0FFh           ; buf_id of free buffer
21298
21299                                 ;Flag byte masks
21300                                 buf_isnet   EQU     10000000B
21301                                 buf_dirty   EQU     01000000B
21302                                 ;***
21303                                 buf_visit   EQU     00100000B
21304                                 ;***
21305                                 buf_snbuf   EQU     00010000B
21306
21307                                 buf_isDATA  EQU     00001000B
21308                                 buf_isDIR   EQU     00000100B
21309                                 buf_isFAT   EQU     00000010B
21310                                 buf_type_0  EQU     11110001B      ; AND sets type to "none"
21311
21312                                 buf_NetID   EQU     bufinsiz
21313
21314                                 ; ----------------------------------------------------------------------------
21315
21316                                 ; ----------------------------------------------------------------------------
21317                                 ;** DPB - Drive Parameter Block
21318
21319                                 ; 25/03/2019 - Retro DOS v4.0 (MSDOS 6.0, DPB.INC, 1991)
21320
21321                                 ;   BUGBUG - this isn't authorative - it's my probably incomplete and
```

```
21322                                  ;   possibly inaccurate deductions from code study... - jgl
21323                                  ;
21324                                  ;   The DPB is DOS's main structure for describing block devices.
21325                                  ;   It contains info about the "Drive" intermingled with info about
21326                                  ;   the FAT file system which is presumably on the drive.  I don't know
21327                                  ;   how those fields are used if it's not the FAT file system - BUGBUG
21328                                  ;
21329                                  ;   The DPBs are statically allocated and chained off of DPBHead.
21330                                  ;   Users scan this chain looking for a match on DPB_DRIVE.
21331                                  ;   The DPBs are built at init time from info in the SYSDEV structure.
21332                                  
21333                                  ; 08/06/2018 - Retro DOS v3.0 (MSDOS 3.3, DPB.INC, 24/07/1987)
21334                                  
21335                                  ; 12/05/2019 - Retro DOS v4.0
21336                                  
21337                                  ; 01/01/2024
21338                                  %if 0
21339                                  
21340                                  struc      DPB
21341                                  .DRIVE:            resb 1         ; Logical drive # assoc with DPB (A=0,B=1,...)
21342                                  .UNIT:             resb 1         ; Driver unit number of DPB
21343                                  .SECTOR_SIZE:      resw 1         ; Size of physical sector in bytes
21344                                  .CLUSTER_MASK:     resb 1         ; Sectors/cluster - 1
21345                                  .CLUSTER_SHIFT:    resb 1         ; Log2 of sectors/cluster
21346                                  .FIRST_FAT: resw 1        ; Starting record of FATs
21347                                  .FAT_COUNT: resb 1        ; Number of FATs for this drive
21348                                  .ROOT_ENTRIES:     resw 1         ; Number of directory entries
21349                                  .FIRST_SECTOR:     resw 1         ; First sector of first cluster
21350                                  .MAX_CLUSTER:      resw 1         ; Number of clusters on drive + 1
21351                                  ; MSDOS 3.3
21352                                  ;.FAT_SIZE: resb 1         ; Number of records occupied by FAT
21353                                  ; MSDOS 6.0
21354                                  .FAT_SIZE:  resw 1         ; Number of records occupied by FAT
21355                                  .DIR_SECTOR:       resw 1         ; Starting record of directory
21356                                  .DRIVER_ADDR:      resd 1         ; Pointer to driver
21357                                  .MEDIA:            resb 1         ; Media byte
21358                                  .FIRST_ACCESS:     resb 1         ; This is initialized to -1 to force a media
21359                                                            ; check the first time this DPB is used
21360                                  .NEXT_DPB:  resd 1         ; Pointer to next Drive parameter block
21361                                  .NEXT_FREE: resw 1         ; Cluster # of last allocated cluster
21362                                  .FREE_CNT:  resw 1         ; Count of free clusters, -1 if unknown
21363                                  .size:
21364                                  endstruc
21365                                  
21366                                  %else
21367                                  
21368                                  ; 01/01/2024 - Retro DOS v5.0 (PCDOS 7.1)
21369                                  
21370                                  struc      DPB
21371 00000000 ??                     .DRIVE:            resb 1  ; 0     ; Logical drive # assoc with DPB (A=0,B=1,...)
21372 00000001 ??                     .UNIT:             resb 1  ; 1     ; Driver unit number of DPB
21373 00000002 ????                   .SECTOR_SIZE:      resw 1  ; 2     ; Size of physical sector in bytes
21374 00000004 ??                     .CLUSTER_MASK:     resb 1  ; 4     ; Sectors/cluster - 1
21375 00000005 ??                     .CLUSTER_SHIFT:    resb 1  ; 5     ; Log2 of sectors/cluster
21376 00000006 ????                   .FIRST_FAT: resw 1  ; 6     ; Starting record of FATs
21377 00000008 ??                     .FAT_COUNT: resb 1  ; 8     ; Number of FATs for this drive
21378 00000009 ????                   .ROOT_ENTRIES:     resw 1  ; 9     ; Number of directory entries
21379 0000000B ????                   .FIRST_SECTOR:     resw 1  ; 11    ; First sector of first cluster
21380 0000000D ????                   .MAX_CLUSTER:      resw 1  ; 13    ; Number of clusters on drive + 1
21381 0000000F ????                   .FAT_SIZE:  resw 1  ; 15    ; Number of records occupied by FAT
21382 00000011 ????                   .DIR_SECTOR:       resw 1  ; 17    ; Starting record of directory
21383 00000013 ????????               .DRIVER_ADDR:      resd 1  ; 19    ; Pointer to driver
21384 00000017 ??                     .MEDIA:            resb 1  ; 23    ; Media byte
21385 00000018 ??                     .FIRST_ACCESS:     resb 1  ; 24    ; This is initialized to -1 to force a media
21386                                                                  ; check the first time this DPB is used
21387 00000019 ????????               .NEXT_DPB:  resd 1  ; 25    ; Pointer to next Drive parameter block
21388 0000001D ????                   .NEXT_FREE: resw 1  ; 29    ; Cluster # of last allocated cluster
21389 0000001F ????                   .FREE_CNT:  resw 1  ; 31    ; Count of free clusters, -1 if unknown
21390                                  ; FAT32 fs ; 01/01/2024
21391                                  ; ref: https://en.wikibooks.org/wiki/
21392                                  ;      First_steps_towards_system_programming_under_MS-DOS_7/Appendix
21393                                  ;   -- A.03-1. Structure of Drive Parameters Blocks (DPB) ---
21394 00000021 ????                   .FREE_CNT_HW:      resw 1  ; 33    ; High word of free cluster count
21395 00000023 ????                   .EXT_FLAGS: resw 1  ; 35    ; FAT32 extended flags (active FAT number)
21396 00000025 ????                   .FSINFO_SECTOR:    resw 1  ; 37    ; (FAT32 fs) FSINFO structure sector address
21397 00000027 ????                   .BKBOOT_SECTOR:    resw 1  ; 39    ; (FAT32 fs) Backup Boot Sector address
21398 00000029 ????????               .FCLUS_FSECTOR: resd 1     ; 41    ; The first cluster's first sector address
21399 0000002D ????????               .LAST_CLUSTER:     resd 1  ; 45    ; The last cluster number
21400 00000031 ????????               .FAT32_SIZE:       resd 1  ; 49    ; Number of FAT sectors (for FAT32 fs)
21401 00000035 ????????               .ROOT_CLUSTER:     resd 1  ; 53    ; Root directory's cluster number (FAT32 fs)
21402                                  ; 01/01/2024 - Retro DOS v5.0
21403 00000039 ????????               .FAT32_NXTFREE:    resd 1  ; 57    ; The next free cluster (for FAT32 fs)
21404                                  .size:             ; 61 bytes ; 01/01/2024 (PCDOS 7.1)
21405                                  endstruc
21406                                  
21407                                  %endif
21408                                  
21409                                  DPBSIZ  EQU     DPB.size  ; Size of the structure in bytes
21410                                  
21411                                  DSKSIZ  EQU DPB.MAX_CLUSTER       ; Size of disk (temp used during init only)
21412                                  
21413                                  ; --------------------------------------------------------------------------
21414                                  ; 26/03/2018
21415                                  
21416                                  ; IOCTL SUB-FUNCTIONS
21417                                  IOCTL_GET_DEVICE_INFO     EQU     0
21418                                  IOCTL_SET_DEVICE_INFO     EQU     1
21419                                  IOCTL_READ_HANDLE  EQU     2
21420                                  IOCTL_WRITE_HANDLE EQU     3
21421                                  IOCTL_READ_DRIVE   EQU     4
21422                                  IOCTL_WRITE_DRIVE  EQU     5
21423                                  IOCTL_GET_INPUT_STATUS    EQU     6
21424                                  IOCTL_GET_OUTPUT_STATUS EQU       7
21425                                  IOCTL_CHANGEABLE?  EQU     8
21426                                  IOCTL_SHARING_RETRY       EQU     11
21427                                  GENERIC_IOCTL_HANDLE      EQU     12
21428                                  GENERIC_IOCTL             EQU     13
21429                                  
21430                                  ; GENERIC IOCTL SUB-FUNCTIONS
21431                                  RAWIO                     EQU     8
21432                                  
21433                                  ; RAWIO SUB-FUNCTIONS
21434                                  GET_DEVICE_PARAMETERS     EQU     60H
21435                                  SET_DEVICE_PARAMETERS     EQU     40H
21436                                  READ_TRACK         EQU     61H
21437                                  WRITE_TRACK        EQU     41H
21438                                  VERIFY_TRACK              EQU     62H
21439                                  FORMAT_TRACK              EQU     42H
21440                                  
21441                                  ; DEVICETYPE VALUES
21442                                  MAX_SECTORS_IN_TRACK      EQU     63
21443                                  DEV_5INCH          EQU     0
21444                                  DEV_5INCH96TPI            EQU     1
21445                                  DEV_3INCH720KB            EQU     2
```

```
21446                                   DEV_8INCHSS         EQU     3
21447                                   DEV_8INCHDS         EQU     4
21448                                   DEV_HARDDISK                EQU     5
21449                                   DEV_OTHER           EQU     7
21450                                   ;DEV_3INCH1440KB    EQU     7
21451                                   DEV_3INCH2880KB             EQU     9
21452                                   ; Retro DOS v2.0 - 26/03/2018
21453                                   ;;DEV_TAPE          EQU     6
21454                                   ;;DEV_ERIMO         EQU     8
21455                                   ;DEV_3INCH2880KB    EQU     9
21456                                   DEV_3INCH1440KB             EQU     10
21457
21458                                   ;MAX_DEV_TYPE               EQU     9       ; MAXIMUM DEVICE TYPE THAT WE
21459                                                               ; CURRENTLY SUPPORT.
21460                                   MAX_DEV_TYPE                EQU     10
21461
21462                                   struc A_SECTORTABLE
21463 00000000 ????                     .ST_SECTORNUMBER:  resw    1
21464 00000002 ????                     .ST_SECTORSIZE:             resw    1
21465                                   .size:
21466                                   endstruc
21467
21468                                   ; 25/03/2019 - Retro DOS v4.0  (MSDOS 6.0, BPB.INC, IOCTL.INC)
21469
21470                                   ;** BIOS PARAMETER BLOCK DEFINITION
21471                                   ;
21472                                   ;   The BPB contains information about the disk structure.  It dates
21473                                   ;   back to the earliest FAT systems and so FAT information is
21474                                   ;   intermingled with physical driver information.
21475                                   ;
21476                                   ;   A boot sector contains a BPB for its device; for other disks
21477                                   ;   the driver creates a BPB.  DOS keeps copies of some of this
21478                                   ;   information in the DPB.
21479                                   ;
21480                                   ;   The BDS structure contains a BPB within it.
21481
21482                                   ; 01/01/2024
21483                                   %if 0
21484
21485                                   struc A_BPB
21486                                   .BPB_BYTESPERSECTOR:       resw    1
21487                                   .BPB_SECTORSPERCLUSTER:    resb    1
21488                                   .BPB_RESERVEDSECTORS:      resw    1
21489                                   .BPB_NUMBEROFFATS: resb    1
21490                                   .BPB_ROOTENTRIES:  resw    1
21491                                   .BPB_TOTALSECTORS: resw    1
21492                                   .BPB_MEDIADESCRIPTOR:      resb    1
21493                                   .BPB_SECTORSPERFAT:        resw    1
21494                                   .BPB_SECTORSPERTRACK:      resw    1
21495                                   .BPB_HEADS:        resw    1
21496                                   .BPB_HIDDENSECTORS:        resw    1
21497                                                      resw    1
21498                                   .BPB_BIGTOTALSECTORS:      resw    1
21499                                                      resw    1
21500                                                      resb    6       ; NOTE:  many times these
21501                                   ;                                  ;       6 bytes are omitted
21502                                   ;                                  ;       when BPB manipulations
21503                                   ;                                  ;       are performed!
21504                                   .size:
21505                                   endstruc
21506
21507                                   %else
21508
21509                                   ; 14/04/2024
21510                                   ; 01/01/2024 - Retro DOS v5.0
21511
21512                                   struc A_BPB
21513 00000000 ????                     .BYTESPERSECTOR:    resw 1
21514 00000002 ??                       .SECTORSPERCLUSTER: resb 1
21515 00000003 ????                     .RESERVEDSECTORS:   resw 1
21516 00000005 ??                       .NUMBEROFFATS:         resb 1
21517 00000006 ????                     .ROOTENTRIES:          resw 1
21518 00000008 ????                     .TOTALSECTORS:         resw 1
21519 0000000A ??                       .MEDIADESCRIPTOR:   resb 1
21520 0000000B ????                     .SECTORSPERFAT:        resw 1
21521 0000000D ????                     .SECTORSPERTRACK:   resw 1
21522 0000000F ????                     .HEADS:                resw 1
21523 00000011 ????????                 .HIDDENSECTORS:        resd 1
21524 00000015 ????????                 .BIGTOTALSECTORS:   resd 1
21525                                   ;............ FAT32 ......  + 28
21526 00000019 ????????                 .FATSIZE32:     resd 1
21527 0000001D ????                     .EXTFLAGS:      resw 1
21528 0000001F ????                     .FSVER:                resw 1
21529 00000021 ????????                 .ROOTDIRCLUSTER:    resd 1
21530 00000025 ????                     .FSINFOSECTOR:         resw 1  ; (offset from FAT32 bs)
21531 00000027 ????                     .BACKUPBOOTSECTOR:  resw 1  ; (offset from FAT32 bs)
21532 00000029 <res Ch>                 .RESERVEDBYTES:        resb 12 ; (zero bytes)
21533                                                  ; 14/04/2024
21534 00000035 ???????????              .BOOTFILESYSTYPE:      resb 6  ; A_BPB.size must be 59
21535                                   .size:
21536                                   endstruc
21537
21538                                   %endif
21539
21540                                   struc A_DEVICEPARAMETERS
21541 00000000 ??                       .DP_SPECIALFUNCTIONS:      resb    1
21542 00000001 ??                       .DP_DEVICETYPE:            resb    1
21543 00000002 ????                     .DP_DEVICEATTRIBUTES:      resw    1
21544 00000004 ????                     .DP_CYLINDERS:             resw    1
21545 00000006 ??                       .DP_MEDIATYPE:             resb    1
21546 00000007 <res 3Bh>                .DP_BPB:           resb    A_BPB.size
21547 00000042 ????                     .DP_TRACKTABLEENTRIES:     resw    1
21548 00000044 <res FCh>                .DP_SECTORTABLE:   resb    MAX_SECTORS_IN_TRACK * A_SECTORTABLE.size
21549                                   endstruc
21550
21551                                   ; ----------------------------------------------------------------------------
21552                                   ; structure, equates for devmark for mem command.
21553
21554                                   ; 25/03/2019 - Retro DOS v4.0 (MSDOS 6.0, DEVMARK.INC, 1991)
21555
21556                                   struc devmark
21557 00000000 ??                        .id:           resb 1
21558 00000001 ????                     .seg:          resw 1
21559 00000003 ????                     .size:         resw 1
21560 00000005 ??????                   .dum:          resb 3
21561 00000008 ????????????????        .filename: resb 8
21562                                   endstruc
21563
21564                                   devmark_stk equ     'S'
21565                                   devmark_device      equ     'D'
21566                                   devmark_ifs equ     'I'
21567                                   devmark_buf equ     'B'
21568                                   devmark_cds equ     'L' ; lastdrive
21569                                   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
```

```
21694                           GET_INTERRUPT_VECTOR            EQU 53  ; 53     35
21695                           GET_DRIVE_FREESPACE             EQU 54  ; 54     36
21696                           CHAR_OPER                       EQU 55  ; 55     37
21697                           INTERNATIONAL                   EQU 56  ; 56     38
21698                           ;   Directory Group
21699                           MKDIR                           EQU 57  ; 57     39
21700                           RMDIR                           EQU 58  ; 58     3A
21701                           CHDIR                           EQU 59  ; 59     3B
21702                           ;   File Group
21703                           CREAT                           EQU 60  ; 60     3C
21704                           OPEN                            EQU 61  ; 61     3D
21705                           CLOSE                           EQU 62  ; 62     3E
21706                           READ                            EQU 63  ; 63     3F
21707                           WRITE                           EQU 64  ; 64     40
21708                           UNLINK                          EQU 65  ; 65     41
21709                           LSEEK                           EQU 66  ; 66     42
21710                           CHMOD                           EQU 67  ; 67     43
21711                           IOCTL                           EQU 68  ; 68     44
21712                           XDUP                            EQU 69  ; 69     45
21713                           XDUP2                           EQU 70  ; 70     46
21714                           CURRENT_DIR                     EQU 71  ; 71     47
21715                           ;    Memory Group
21716                           ALLOC                           EQU 72  ; 72     48
21717                           DEALLOC                         EQU 73  ; 73     49
21718                           SETBLOCK                        EQU 74  ; 74     4A
21719                           ;    Process Group
21720                           EXEC                            EQU 75  ; 75     4B
21721                           EXIT                            EQU 76  ; 76     4C
21722                           WAITPROCESS                 EQU 77  ; 77     4D
21723                           FIND_FIRST                      EQU 78  ; 78     4E
21724                           ;   Special Group
21725                           FIND_NEXT                       EQU 79  ; 79     4F
21726                           ; SPECIAL SYSTEM GROUP
21727                           SET_CURRENT_PDB                 EQU 80  ; 80     50
21728                           GET_CURRENT_PDB                 EQU 81  ; 81     51
21729                           GET_IN_VARS                     EQU 82  ; 82     52
21730                           SETDPB                          EQU 83  ; 83     53
21731                           GET_VERIFY_ON_WRITE             EQU 84  ; 84     54
21732                           DUP_PDB                         EQU 85  ; 85     55
21733                           RENAME                          EQU 86  ; 86     56
21734                           FILE_TIMES                      EQU 87  ; 87     57
21735                           ;
21736                           ALLOCOPER                   EQU 88  ; 88     58
21737                           ; Network extention system calls
21738                           GetExtendedError            EQU 89  ; 89     59
21739                           CreateTempFile                  EQU 90  ; 90     5A
21740                           CreateNewFile                   EQU 91  ; 91     5B
21741                           LockOper                    EQU 92  ; 92     5C Lock and Unlock
21742                           ServerCall                  EQU 93  ; 93     5D CommitAll, ServerDOSCall,
21743                                                               ;           CloseByName, CloseUser,
21744                                                               ;           CloseUserProcess,
21745                                                               ;           GetOpenFileList
21746                           UserOper                    EQU 94  ; 94     5E Get and Set
21747                           AssignOper                  EQU 95  ; 95     5F On, Off, Get, Set, Cancel
21748                           xNameTrans                  EQU 96  ; 96     60
21749                           PathParse                   EQU 97  ; 97     61
21750                           GetCurrentPSP                   EQU 98  ; 98     62
21751                           Hongeul                     EQU 99  ; 99     63
21752                           ECS_CALL                    EQU 99  ; 99     63  ;; DBCS support
21753                           Set_Printer_Flag            EQU 100 ; 100    64
21754                           GetExtCntry                 EQU 101 ; 101    65
21755                           GetSetCdPg                  EQU 102 ; 102    66
21756                           ExtHandle                   EQU 103 ; 103    67
21757                           Commit                          EQU 104 ; 104    68
21758                           GetSetMediaID                   EQU 105 ; 105    69
21759                           IFS_IOCTL                   EQU 107 ; 107    6B
21760                           ExtOpen                     EQU 108 ; 108    6C
21761                           ;
21762                           ;ifdef ROMEXEC
21763                           ;ROM_FIND_FIRST                 EQU 109 ; 109     6D
21764                           ;ROM_FIND_NEXT                  EQU 110 ; 110     6E
21765                           ;ROM_EXCLUDE                    EQU 111 ; 111    6F
21766                           ;endif
21767                           ;
21768                           Set_Oem_Handler             EQU 248 ; 248    F8
21769                           OEM_C1                          EQU 249 ; 249    F9
21770                           OEM_C2                          EQU 250 ; 250    FA
21771                           OEM_C3                          EQU 251 ; 251    FB
21772                           OEM_C4                          EQU 252 ; 252    FC
21773                           OEM_C5                          EQU 253 ; 253    FD
21774                           OEM_C6                          EQU 254 ; 254    FE
21775                           OEM_C7                          EQU 255 ; 255    FF
21776
21777                           ; ----------------------------------------------------------------------
21778                           ; SYSCONF.ASM (MSDOS 3.3 - 24/07/1987)
21779                           ; ----------------------------------------------------------------------
21780
21781                           ;;  IF      STACKSW
21782
21783                           ;;
21784                           ;; Internal Stack Parameters
21785                           ;EntrySize          equ     8
21786                           ;
21787                           ;MinCount           equ     8
21788                           ;DefaultCount               equ     9
21789                           ;MaxCount           equ     64
21790                           ;
21791                           ;MinSize            equ     32
21792                           ;DefaultSize                equ     128
21793                           ;MaxSize            equ     512
21794
21795                           ;;  ENDIF
21796
21797                           ; ----------------------------------------------------------------------
21798                           ; BIOSTRUC.INC (MSDOS 3.3 - 24/07/1987)
21799                           ; ----------------------------------------------------------------------
21800                                                   ;;Rev 3.30 Modification
21801                           ; ROM BIOS CALL PACKET STRUCTURES
21802
21803                           ;*********************************
21804                           ;System Service call ( Int 15h )
21805                           ;*********************************
21806                           ;Function AH = 0C0h, Return system configuration
21807                           ;For PC and PCJR on return:
21808                           ;   (AH)    = 80h
21809                           ;   (CY)    = 1
21810                           ;For PCXT, PC PORTABLE and PCAT on return:
21811                           ;   (AH)    = 86h
21812                           ;   (CY)    = 1
21813                           ;For all others:
21814                           ;   (AH)    = 0
21815                           ;   (CY)    = 0
21816                           ;   (ES:BX) = pointer to system descriptor vector in ROS
21817                           ; System descriptor :
```

```
21818                                   ;   DW      xxxx            length of descriptor in bytes,
21819                                   ;                           minimum length = 8
21820                                   ;   DB      xx              model byte
21821                                   ;                           0FFh    = PC
21822                                   ;                           0FEh    = PC/XT, Portable
21823                                   ;                           0FDh    = PC/JR
21824                                   ;                           0FCh    = PC/AT
21825                                   ;                           0F9h    = Convertable
21826                                   ;                           0F8h    = Model 80
21827                                   ;                           0E0 thru 0EFh = reserved
21828                                   ;
21829                                   ;   DB      xx              secondary model byte
21830                                   ;                           000h    = PC1
21831                                   ;                           000h    = PC/XT, Portable
21832                                   ;                           000h    = PC/JR
21833                                   ;                           000h    = PC/AT
21834                                   ;                           001h    = PC/AT Model 339
21835                                   ;                           003h    = PC/RT
21836                                   ;                           000h    = Convertable
21837                                   ;
21838                                   ;   DB      xx              bios revision level
21839                                   ;                           00 for first release, subsequent release
21840                                   ;                           of code with same model byte and
21841                                   ;                           secondary model byte require revison level
21842                                   ;                           to increase by one.
21843                                   ;
21844                                   ;   DB      xx              feature information byte 1
21845                                   ;                           X0000000 = 1, bios use DMA channel 3
21846                                   ;                                    = 0, DMA channel 3 not used
21847                                   ;
21848                                   ;                           0X000000 = 1, 2nd Interrupt chip present
21849                                   ;                                    = 0, 2nd Interrupt chip not present
21850                                   ;
21851                                   ;                           00X00000 = 1, Real Time Clock present
21852                                   ;                                    = 0, Real Time Clock not present
21853                                   ;
21854                                   ;                           000X0000 = 1, Keyboard escape sequence(INT 15h)
21855                                   ;                                        called in keyboard interrupt
21856                                   ;                                        (Int 09h).
21857                                   ;                                  = 0, Keyboard escape sequence not
21858                                   ;                                        called.
21859                                   ;                                0000XXXX reserved
21860                                   ;
21861                                   ;   DB      xx              feature information byte 2 - reserved
21862                                   ;
21863                                   ;   DB      xx              feature information byte 2 - reserved
21864                                   ;
21865                                   ;   DB      xx              feature information byte 2 - reserved
21866                                   ;
21867                                   ;   DB      xx              feature information byte 2 - reserved
21868                                   ;
21869
21870                                   ; 22/03/2019
21871                                   struc ROMBIOS_DESC          ; BIOS_SYSTEM_DESCRIPTOR
21872 00000000 ????                    .bios_sd_leng:              resw 1
21873 00000002 ??                      .bios_sd_modelbyte:         resb 1
21874                                   .bios_sd_scnd_modelbyte:
21875 00000003 ??                                  resb 1
21876 00000004 ??                                  resb 1
21877 00000005 ??                      .bios_sd_featurebyte1:      resb 1
21878 00000006 ????????                           resb 4
21879                                   endstruc
21880
21881                                   ;FeatureByte1       bit map equates
21882                                   DMAchannel3         equ 10000000b
21883                                   ScndIntController   equ 01000000b
21884                                   RealTimeClock               equ 00100000b
21885                                   KeyEscapeSeq                equ 00010000b
21886                                                                   ;;End of Modification
21887
21888                                   ; ------------------------------------------------------------------------
21889                                   ; SYSVAR.INC (MSDOS 6.0 - 1991)
21890                                   ; ------------------------------------------------------------------------
21891                                   ; 22/03/2019 - Retro DOS v4.0
21892
21893                                   ;   SCCSID = @(#)sysvar.asm 1.1 85/04/10
21894
21895                                   struc SysInitVars
21896                                   ; MSDOS 3.3
21897 00000000 ????????                .SYSI_DPB:      resd 1                  ; DPB chain
21898 00000004 ????????                .SYSI_SFT:      resd 1                  ; SFT chain
21899 00000008 ????????                .SYSI_CLOCK:    resd 1                  ; CLOCK device
21900 0000000C ????????                .SYSI_CON:      resd 1                  ; CON device
21901 00000010 ????                    .SYSI_MAXSEC:   resw 1                  ; maximum sector size
21902 00000012 ????????                .SYSI_BUF:      resd 1                  ; buffer chain
21903 00000016 ????????                .SYSI_CDS:      resd 1                  ; CDS list
21904 0000001A ????????                .SYSI_FCB:      resd 1                  ; FCB chain
21905 0000001E ????                    .SYSI_KEEP:     resw 1                  ; keep count
21906 00000020 ??                      .SYSI_NUMIO:    resb 1                  ; number of block devices
21907 00000021 ??                      .SYSI_NCDS:     resb 1                  ; number of CDS's
21908 00000022 ????????                .SYSI_DEV:      resd 1                  ; device list
21909                                   ; MSDOS 6.0
21910 00000026 ????                    .SYSI_ATTR:     resw 1          ; null device attribute word
21911 00000028 ????                    .SYSI_STRAT:            resw 1          ; null device strategy entry point
21912 0000002A ????                    .SYSI_INTER:            resw 1          ; null device interrupt entry point
21913 0000002C ????????????????        .SYSI_NAME:     resb 8          ; null device name
21914                                   .SYSI_SPLICE:           resb 0          ; TRUE -> splicees being done
21915 00000034 ????                    .SYSI_IBMDOS_SIZE:  resw 1      ; DOS size in paragraphs
21916 00000036 ????????                .SYSI_IFS_DOSCALL@: resd 1      ; IFS DOS service rountine entry
21917 0000003A ????????                .SYSI_IFS:      resd 1          ; IFS header chain
21918 0000003E ????????                .SYSI_BUFFERS:          resw 2          ; BUFFERS= values (m,n)
21919 00000042 ??                      .SYSI_BOOT_DRIVE:   resb 1      ; boot drive A=1 B=2,..
21920 00000043 ??                      .SYSI_DWMOVE:           resb 1          ; 1 if 386 machine
21921 00000044 ????                    .SYSI_EXT_MEM:          resw 1          ; Extended memory size in KB.
21922                                   .size:
21923                                   endstruc
21924
21925                                   ;This is added for more information exchage between DOS, BIOS.
21926                                   ;DOS will give the pointer to SysInitTable in ES:DI. - J.K. 5/29/86
21927
21928                                   ; 22/03/2019
21929                                   struc SysInitVars_Ext
21930 00000000 ????????                .SYSI_InitVars:       resd 1    ; Points to the above structure.
21931 00000004 ????????                .SYSI_Country_Tab: resd 1 ; DOS_Country_cdpg_info
21932                                   endstruc
21933
21934                                   ; 09/06/2018
21935                                   ; 08/06/2018 - Retro DOS v3.0 (MSDOS 3.3)
21936                                   SYSI_DPB    equ   0
21937                                   SYSI_SFT    equ 4
21938                                   SYSI_CLOCK  equ 8
21939                                   SYSI_CON    equ 12
21940                                   SYSI_MAXSEC equ 16
21941                                   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
```

```
22066 00000004 ????                    .SFCount:   resw    1             ; number of entries
22067 00000006 ????                    .SFTable:   resw    1             ; beginning of array of the following
22068                                  .size:
22069                                  endstruc
22070
22071                                  ;** System file table entry
22072                                  ;
22073                                  ;   These are the structures which are at SFTABLE in the SF structure.
22074
22075                                  struc       SF_ENTRY
22076 00000000 ????                    .sf_ref_count:     resw    1              ; number of processes sharing entry
22077                                                                        ;   if FCB then ref count
22078 00000002 ????                    .sf_mode:   resw    1             ; mode of access or high bit on if FCB
22079 00000004 ??                      .sf_attr:   resb    1             ; attribute of file
22080 00000005 ????                    .sf_flags:  resw    1             ;Bits 8-15
22081                                                                        ; Bit 15 = 1 if remote file
22082                                                                        ;       = 0 if local file or device
22083                                                                        ; Bit 14 = 1 if date/time is not to be
22084                                                                        ;   set from clock at CLOSE.  Set by
22085                                                                        ;   FILETIMES and FCB_CLOSE.  Reset by
22086                                                                        ;   other reseters of the dirty bit
22087                                                                        ;   (WRITE)
22088                                                                        ; Bit 13 = Pipe bit (reserved)
22089                                                                        ;
22090                                                                        ; Bits 0-7 (old FCB_devid bits)
22091                                                                        ; If remote file or local file, bit
22092                                                                        ; 6=0 if dirty Device ID number, bits
22093                                                                        ; 0-5 if local file.
22094                                                                        ; bit 7=0 for local file, bit 7
22095                                                                        ;       =1 for local I/O device
22096                                                                        ; If local I/O device, bit 6=0 if EOF (input)
22097                                                                        ;             Bit 5=1 if Raw mode
22098                                                                        ;             Bit 0=1 if console input device
22099                                                                        ;             Bit 1=1 if console output device
22100                                                                        ;             Bit 2=1 if null device
22101                                                                        ;             Bit 3=1 if clock device
22102 00000007 ????????                .sf_devptr: resd    1             ; Points to DPB if local file, points
22103                                                                        ; to device header if local device,
22104                                                                        ; points to net device header if
22105                                                                        ; remote
22106 0000000B ????                    .sf_firclus:       resw    1              ; First cluster of file (bit 15 = 0)
22107                                  ;.sf_lstclus:      resw    1 ; *
22108 0000000D ????                    .sf_time:   resw    1             ; Time associated with file
22109 0000000F ????                    .sf_date:   resw    1             ; Date associated with file
22110 00000011 ????????                .sf_size:   resd    1             ; Size associated with file
22111 00000015 ????????                .sf_position:      resd    1              ; Read/Write pointer or LRU count for FCBs
22112                                  ;
22113                                  ; Starting here, the next 7 bytes may be used by the file system to store an
22114                                  ; ID
22115                                  ;
22116 00000019 ????                    .sf_cluspos:       resw    1              ; Position of last cluster accessed
22117 0000001B ????????                .sf_dirsec: resd    1 ; 09/04/2024 ; Sector number of directory sector for this file
22118 0000001F ??                      .sf_dirpos: resb    1             ; Offset of this entry in the above
22119                                  ;
22120                                  ; End of 7 bytes of file-system specific info.
22121                                  ;
22122 00000020 <res Bh>                .sf_name:   resb    11            ; 11 character name that is in the
22123                                                                        ; directory entry.  This is used by
22124                                                                        ; close to detect file deleted and
22125                                                                        ; disk changed errors.
22126                                  ; SHARING INFO
22127 0000002B ????????                .sf_chain:  resd    1             ; link to next SF
22128 0000002F ????                    .sf_UID:    resw    1
22129 00000031 ????                    .sf_PID:    resw    1
22130 00000033 ????                    .sf_MFT:    resw    1
22131 00000035 ????                    .sf_lstclus:       resw    1 ; *          ; Last cluster accessed
22132 00000037 ????????                .sf_IFS_HDR:       resd    1 ; **
22133                                  .size:
22134                                  endstruc
22135
22136                                  ; ----------------------------------------------------------------------
22137                                  ; DOSCNTRY.INC (MSDOS 3.3 - 24/07/1987)
22138                                  ; ----------------------------------------------------------------------
22139                                  ; 11/06/2018 - Retro DOS v3.0
22140
22141                                  ;Equates for COUNTRY INFORMATION.
22142                                  SetCountryInfo             EQU     1       ;country info
22143                                  SetUcase           EQU     2       ;uppercase table
22144                                  SetLcase           EQU     3       ;lowercase table (Reserved)
22145                                  SetUcaseFile               EQU     4       ;uppercase file spec table
22146                                  SetFileList        EQU     5       ;valid file character list
22147                                  SetCollate         EQU     6       ;collating sequence
22148                                  SetDBCS            EQU     7       ;double byte character set
22149                                  SetALL                     EQU     -1      ;all the entries
22150
22151                                  ;DOS country and code page information table structure.
22152                                  ;Internally, IBMDOS gives a pointer to this table.
22153                                  ;IBMBIO, MODE and NLSFUNC modules communicate with IBMDOS through
22154                                  ;this structure.
22155
22156                                  struc country_cdpg_info ; DOS_country_cdpg_info
22157 00000000 ??????????????????          .ccInfo_reserved : resb    8       ;reserved for internal use
22158 00000008 <res 40h>               .ccPath_CountrySys:        resb    64      ;path and filename for country info
22159 00000048 ????                    .ccSysCodePage:            resw    1       ;system code page id
22160 0000004A ????                    .ccNumber_of_entries:      resw    1 ; dw 5
22161 0000004C ??                      .ccSetUcase:               resb    1 ; db SetUcase ; = 2
22162 0000004D ????????                .ccUcase_ptr:              resd    1       ;pointer to Ucase table
22163
22164 00000051 ??                      .ccSetUcaseFile:   resb    1 ; db SetUcaseFile ; = 4
22165 00000052 ????????                .ccFileUcase_ptr:  resd    1       ;pointer to File Ucase table
22166
22167 00000056 ??                      .ccSetFileList:            resb    1 ; db SetFileList ; = 5
22168 00000057 ????????                .ccFileChar_ptr:   resd    1       ;pointer to File char list table
22169
22170 0000005B ??                      .ccSetCollate:             resb    1 ; db SetCollate ; = 6
22171 0000005C ????????                .ccCollate_ptr:            resd    1       ;pointer to collate table
22172
22173 00000060 ??                      .ccSetCountryInfo: resb    1 ; db SetCountryInfo ; = 1
22174 00000061 ????                    .ccCountryInfoLen: resw    1       ;length of country info
22175 00000063 ????                    .ccDosCountry:             resw    1       ;system country code id
22176 00000065 ????                    .ccDosCodePage:            resw    1       ;system code page id
22177 00000067 ????                    .ccDFormat:        resw    1       ;date format
22178 00000069 ??????????              .ccCurSymbol:              resb    5 ; db "    ",0
22179                                                                        ;5 byte of (currency symbol+0)
22180 0000006E ????                    .cc1000Sep:        resb    2 ; db " ",0 ;2 byte of (1000 sep. + 0)
22181 00000070 ????                    .ccDecSep:         resb    2 ; db " ",0 ;2 byte of (Decimal sep. + 0)
22182 00000072 ????                    .ccDateSep:        resb    2 ; db " ",0 ;2 byte of (date sep. + 0)
22183 00000074 ????                    .ccTimeSep:        resb    2 ; db " ",0 ;2 byte of (time sep. + 0)
22184 00000076 ??                      .ccCFormat:        resb    1       ;currency format flags
22185 00000077 ??                      .ccCSigDigits:             resb    1       ;# of digits in currency
22186 00000078 ??                      .ccTFormat:        resb    1       ;time format
22187 00000079 ????????                .ccMono_Ptr:               resd    1       ;monocase routine entry point
22188 0000007D ????                    .ccListSep:        resb    2 ; db " ",0 ;data list separator
22189 0000007F <res Ah>                .ccReserved_area:  resw    5 ; dw 5 dup(?) ;reserved
```

```
22190                           .size:
22191                           endstruc
22192                           
22193                           NEW_COUNTRY_SIZE    equ  country_cdpg_info.size - country_cdpg_info.ccDosCountry
22194                           
22195                           ; ==========================================================================
22196                           ; retrodos4.s (offset addresses in MSDOS.SYS or RETRODOS.SYS)
22197                           ; ==========================================================================
22198                           ; 21/03/2019 - Retro DOS v4.0
22199                           ; 21/10/2022 - Retro DOS v4.0 (MOdified MSDOS 5.0 IO.SYS)
22200                           
22201                           ;KERNEL_SEGMENT    equ 0070h  ; (IO.SYS loading segment, BIOS_DATA segment)
22202                           ; 21/10/2022
22203                           DOSBIODATASEG equ 0070h    ; (IO.SYS loading segment, BIOS_DATA segment)
22204                           ; 22/10/2022
22205                           ;DOSBIOCODESEG equ 02C7h ; (MSDOS 5.0 IO.SYS, BIOS_CODE segment)
22206                           ; 09/12/2022
22207                           DOSBIOCODESEG equ IOSYSCODESEG
22208                           
22209                           ; Note: These offset addresses must be changed when the code
22210                           ;   in retrodos4.s (MSDOS.SYS) file will be changed.
22211                           
22212                           ; (following addresses can be verified by searching them in retrodos4.lst)
22213                           
22214                           ; 09/12/2022
22215                           %if 0
22216                           
22217                           ; 13/05/2019
22218                           
22219                           ;IsWin386         equ 08CFh
22220                           ;V86_Crit_SetFocus equ 08D0h
22221                           ; 21/10/2022
22222                           IsWin386          equ 08D0h
22223                           V86_Crit_SetFocus equ 08D1h
22224                           
22225                           ;seg_reinit   equ 0772h ; not used in Retro DOS v4.0
22226                           ; 21/10/2022 - Retro DOS v4.0 (MOdified MSDOS 5.0 IO.SYS)
22227                           seg_reinit    equ 0032h ; DOSBIOCODESEG:0032h
22228                           
22229                           ;SysinitPresent      equ 08FCh
22230                           ; 21/10/2022
22231                           SysinitPresent       equ 08FDh
22232                           
22233                           inHMA                equ 000Dh
22234                           xms           equ 000Eh
22235                           ;FreeHMAPtr   equ 08F6h
22236                           ;multrk_flag         equ 0533h
22237                           ;ec35_flag    equ 0535h
22238                           ;EOT          equ 012Eh
22239                           ; 21/10/2022
22240                           FreeHMAPtr    equ 08F7h
22241                           multrk_flag   equ 052Fh
22242                           ec35_flag     equ 0531h
22243                           EOT           equ 012Ch
22244                           
22245                           ;NextStack    equ 08BFh
22246                           ;IT_StackLoc         equ 08C5h
22247                           ;IT_StackSize        equ 08C9h
22248                           ; 21/10/2022
22249                           NextStack     equ 08C0h
22250                           IT_StackLoc   equ 08C6h
22251                           IT_StackSize         equ 08CAh
22252                           
22253                           ;MoveDOSIntoHMA      equ 08F8h
22254                           ; 21/10/2022
22255                           MoveDOSIntoHMA       equ 08F9h
22256                           
22257                           ;INT19SEM equ 0644h ; 01/05/2019 - retrodos4.lst
22258                           ;I19_LST  equ 0645h ; 27/03/2019 - retrodos4.lst
22259                           ; 21/10/2022
22260                           INT19SEM equ 0640h ; (iosys5.txt)
22261                           I19_LST  equ 0641h ; (iosys5.txt)
22262                           
22263                           %endif
22264                           
22265                           ; 09/12/2022
22266                           seg_reinit equ _seg_reinit
22267                           ec35_flag  equ ec35flag
22268                           INT19SEM   equ int19sem
22269                           I19_LST    equ i19_lst
22270                           
22271                           INT19OLD02 equ I19_LST+1 ; 0642h ; 21/10/2022
22272                           INT19OLD08 equ I19_LST+6
22273                           INT19OLD09 equ I19_LST+11
22274                           INT19OLD0A equ I19_LST+16
22275                           INT19OLD0B equ I19_LST+21
22276                           INT19OLD0C equ I19_LST+26
22277                           INT19OLD0D equ I19_LST+31
22278                           INT19OLD0E equ I19_LST+36
22279                           INT19OLD70 equ I19_LST+41
22280                           INT19OLD72 equ I19_LST+46
22281                           INT19OLD73 equ I19_LST+51
22282                           INT19OLD74 equ I19_LST+56
22283                           INT19OLD76 equ I19_LST+61
22284                           INT19OLD77 equ I19_LST+66 ; 0683h ; 21/10/2022
22285                           
22286                           ; 09/12/2022
22287                           %if 0
22288                           
22289                           ;keyrd_func equ 04E9h
22290                           ;keysts_func       equ 04EAh
22291                           ;t_switch   equ 04F6h
22292                           ; 21/10/2022
22293                           keyrd_func  equ 04E5h
22294                           keysts_func equ 04E6h
22295                           t_switch    equ 04F2h
22296                           
22297                           ; 22/10/2022
22298                           SYSINITSEG  equ 046Dh  ; SYSINIT segment
22299                           BCODE_END   equ (SYSINITSEG-DOSBIOCODESEG)*16 ; = 1A60h
22300                           BCODE_START equ 30h  ; (offset BiosDataWord in DOSBIOCODESEG)
22301                           RE_INIT            equ 089Bh ; (re_init offset in DOSBIODATASEG)
22302                           
22303                           %endif
22304                           
22305                           ; 09/12/2022
22306                           BCODESTART  equ BIOSDATAWORD
22307                           RE_INIT            equ re_init
22308                           
22309                           ; --------------------------------------------------------------------------
22310                           ; CONFIG.INC (MSDOS 6.0 - 1991)
22311                           ; --------------------------------------------------------------------------
22312                           ; 15/04/2019 - Retro DOS v4.0
22313
```

```
                        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
```

```
22438                                   ;   the first (= 80h) or not.  The FirstFlag entry of INT77h's
22439                                   ;   program header is initialized in STKINIT.INC module.
22440                                   ;   FirstFlag is only meaningful for interrupt level 7 and 15.
22441                                   ;
22442                                   
22443                                   ;  User specifies the number of stack elements - default = 9
22444                                   ;                                               minimum = 8
22445                                   ;                                               maximum = 64
22446                                   ;
22447                                   ;  Intercepts Asynchronous Hardware Interrupts only
22448                                   ;
22449                                   ;  Picks a stack from pool of stacks and switches to it
22450                                   ;
22451                                   ;  Calls the previously saved interrupt vector after pushing flags
22452                                   ;
22453                                   ;  On return, returns the stack to the stack pool
22454                                   ;
22455                                   
22456                                   ; This is a modification of STACKS:
22457                                   ; 1. To fix a bug which was causing the program to take up too much space.
22458                                   ; 2. To dispense stack space from hi-mem first rather than low-mem first.
22459                                   ;    . Clobbers the stack that got too big instead of innocent stack
22460                                   ;    . Allows system to work if the only stack that got too big was the most
22461                                   ;      deeply nested one
22462                                   ; 3. Disables NMI interrupts while setting the NMI vector.
22463                                   ; 4. Does not intercept any interrupts on a PCjr.
22464                                   ; 5. Double checks that a nested interrupt didn't get the same stack.
22465                                   ; 6. Intercepts Ints 70, 72-77 for PC-ATs and other future products
22466                                   
22467                                               ;EVEN
22468                                   ;align 2
22469                                               ; 21/10/2022
22470                                   
22471 00000000 0000                               dw      0       ; spare field but leave these in order
22472 00000002 0000                   stackcount: dw      0
22473 00000004 0000                   stackat:    dw      0
22474 00000006 0000                   stacksize:  dw      0
22475 00000008 0000                   stacks:             dw      0
22476 0000000A 0000                               dw      0
22477                                   
22478 0000000C [0800]                 firstentry: dw      stacks
22479 0000000E [4800]                 lastentry:  dw      stacks+(defaultcount*entrysize)-entrysize
22480 00000010 [4800]                 nextentry:  dw      stacks+(defaultcount*entrysize)-entrysize
22481                                   
22482                                   ;*****************************************************************************
22483                                   ; THESE ARE THE INDIVIDUAL INTERRUPT HANDLERS
22484                                   
22485                                   ; ---------------------------------------------------------------------------
22486                                   
22487 00000012 00000000               old02:      dd      0
22488                                   
22489                                   int02:
22490                                   
22491                                   ; ****************************************************************************
22492                                   ;
22493                                   ; this is special support for the pc convertible / nmi handler
22494                                   ;
22495                                   ;   on the pc convertible, there is a situation where an nmi can be
22496                                   ;   caused by using the "out" instructions to certain ports. when this
22497                                   ;   occurs, the pc convertible hardware *guarantees* that **nothing**
22498                                   ;   can stop the nmi or interfere with getting to the nmi handler. this
22499                                   ;   includes other type of interrupts (hardware and software), and
22500                                   ;   also includes other type of nmi's. when any nmi has occured,
22501                                   ;   no other interrtupt (hardware, software or nmi) can occur until
22502                                   ;   the software takes specific steps to allow further interrupting.
22503                                   ;
22504                                   ;   for pc convertible, the situation where the nmi is generated by the
22505                                   ;   "out" to a control port requires "fixing-up" and re-attempting. in
22506                                   ;   otherwords, it is actually a "restartable exception". in this
22507                                   ;   case, the software handler must be able to get to the stack in
22508                                   ;   order to figure out what instruction caused the problem, where
22509                                   ;   it was "out"ing to and what value it was "out"ing.  therefore,
22510                                   ;   we will not switch stacks in this situation. this situation is
22511                                   ;   detected by interrogating port 62h, and checking for a bit value
22512                                   ;   of 80h. if set, *****do not switch stacks*****.
22513                                   ;
22514                                   ; ****************************************************************************
22515                                   
22516 00000016 50                          push    ax
22517 00000017 06                          push    es
22518 00000018 B800F0                      mov     ax,0F000h
22519 0000001B 8EC0                        mov     es,ax
22520                                       ; 02/11/2022
22521 0000001D 26803EFEFFF9                cmp     byte [es:0FFFEh],0F9h ; mdl_convert ; check if convertible
22522 00000023 07                          pop     es
22523 00000024 750C                        jne     short normal02
22524                                   
22525 00000026 E462                        in      al,62h          ; PC/XT PPI port C. Bits:
22526                                                               ; 0-3: values of DIP switches
22527                                                               ; 5: 1=Timer 2 channel out
22528                                                               ; 6: 1=I/O channel check
22529                                                               ; 7: 1=RAM parity check error occurred.
22530 00000028 A880                        test    al,80h
22531 0000002A 7406                        jz      short normal02
22532                                   special02:
22533 0000002C 58                          pop     ax
22534 0000002D 2EFF2E[1200]                jmp     far [cs:old02]
22535                                   normal02:
22536 00000032 58                          pop     ax
22537 00000033 E81101                      call    do_int_stacks
22538 00000036 [1200]                      dw      old02
22539                                   
22540                                   ; ---------------------------------------------------------------------------
22541                                   
22542 00000038 00000000               old08:      dd      0
22543                                   
22544                                   int08:
22545 0000003C E80801                      call    do_int_stacks
22546 0000003F [3800]                      dw      old08
22547                                   
22548                                   ; ---------------------------------------------------------------------------
22549                                   
22550 00000041 00000000               old09:      dd      0
22551                                   
22552                                   int09:
22553                                   
22554                                   ; keyboard interrupt must have a three byte jump, a nop and a zero byte
22555                                   ; as its first instruction for compatibility reasons
22556                                   
22557 00000045 EB02                        jmp     short keyboard_lbl
22558 00000047 90                          nop
22559 00000048 00                          db      0
22560                                   
22561                                   keyboard_lbl:
```

```
22562 00000049 E8FB00                   call    do_int_stacks
22563 0000004C [4100]                   dw      old09
22564
22565                                   ; --------------------------------------------------------------------------
22566
22567 0000004E 00000000                 old70:      dd      0
22568
22569                                   int70:
22570 00000052 E8F200                       call    do_int_stacks
22571 00000055 [4E00]                       dw      old70
22572
22573                                   ; --------------------------------------------------------------------------
22574
22575                                   ;   irp     a,<0a,0b,0c,0d,0e,72,73,74,76,77>
22576                                   ;public     int&a
22577                                   ;public     old&a
22578                                   ;public     firstflag&a
22579                                   ;int&a      proc    far
22580                                   ;   jmp     short entry_int&a&_stk
22581                                   ;old&a      dd       0          ;forward pointer
22582                                   ;   dw       424bh        ;compatible signature for int. sharing
22583                                   ;firstflag&a db   0             ;the firstly hooked.
22584                                   ;   jmp     short intret_&a       ;reset routine. we don't care this.
22585                                   ;   db      7 dup (0)      ;reserved for future.
22586                                   ;entry_int&a&_stk:
22587                                   ;   call    do_int_stacks
22588                                   ;   dw      old&a
22589                                   ;intret_&a:
22590                                   ;   iret
22591                                   ;int&a      endp
22592                                   ;   endm
22593
22594                                   ; --------------------------------------------------------------------------
22595
22596                                   int0A:
22597 00000057 EB10                         jmp     short entry_int0A_stk
22598 00000059 00000000                 old0A:      dd      0
22599 0000005D 4B42                         dw      424Bh
22600                                   firstflag0A:
22601 0000005F 00                           db      0
22602 00000060 EB0C                         jmp     short intret_0A
22603 00000062 00<rep 7h>                   times   7 db 0
22604
22605                                   entry_int0A_stk:
22606 00000069 E8DB00                       call    do_int_stacks
22607 0000006C [5900]                       dw      old0A
22608                                   intret_0A:
22609 0000006E CF                           iret
22610
22611                                   ; --------------------------------------------------------------------------
22612
22613                                   int0B:
22614 0000006F EB10                         jmp     short entry_int0B_stk
22615 00000071 00000000                 old0B:      dd      0
22616 00000075 4B42                         dw      424Bh
22617                                   firstflag0B:
22618 00000077 00                           db      0
22619 00000078 EB0C                         jmp     short intret_0B
22620 0000007A 00<rep 7h>                   times   7 db 0
22621
22622                                   entry_int0B_stk:
22623 00000081 E8C300                       call    do_int_stacks
22624 00000084 [7100]                       dw      old0B
22625                                   intret_0B:
22626 00000086 CF                           iret
22627
22628                                   ; --------------------------------------------------------------------------
22629
22630                                   int0C:
22631 00000087 EB10                         jmp     short entry_int0C_stk
22632 00000089 00000000                 old0C:      dd      0
22633 0000008D 4B42                         dw      424Bh
22634                                   firstflag0C:
22635 0000008F 00                           db      0
22636 00000090 EB0C                         jmp     short intret_0C
22637 00000092 00<rep 7h>                   times   7 db 0
22638
22639                                   entry_int0C_stk:
22640 00000099 E8AB00                       call    do_int_stacks
22641 0000009C [8900]                       dw      old0C
22642                                   intret_0C:
22643 0000009E CF                           iret
22644
22645                                   ; --------------------------------------------------------------------------
22646
22647                                   int0D:
22648 0000009F EB10                         jmp     short entry_int0D_stk
22649 000000A1 00000000                 old0D:      dd      0
22650 000000A5 4B42                         dw      424Bh
22651                                   firstflag0D:
22652 000000A7 00                           db      0
22653 000000A8 EB0C                         jmp     short intret_0D
22654 000000AA 00<rep 7h>                   times   7 db 0
22655
22656                                   entry_int0D_stk:
22657 000000B1 E89300                       call    do_int_stacks
22658 000000B4 [A100]                       dw      old0D
22659                                   intret_0D:
22660 000000B6 CF                           iret
22661
22662                                   ; --------------------------------------------------------------------------
22663
22664                                   int0E:
22665 000000B7 EB10                         jmp     short entry_int0E_stk
22666 000000B9 00000000                 old0E:      dd      0
22667 000000BD 4B42                         dw      424Bh
22668                                   firstflag0E:
22669 000000BF 00                           db      0
22670 000000C0 EB0C                         jmp     short intret_0E
22671 000000C2 00<rep 7h>                   times   7 db 0
22672
22673                                   entry_int0E_stk:
22674 000000C9 E87B00                       call    do_int_stacks
22675 000000CC [B900]                       dw      old0E
22676                                   intret_0E:
22677 000000CE CF                           iret
22678
22679                                   ; --------------------------------------------------------------------------
22680
22681                                   int72:
22682 000000CF EB10                         jmp     short entry_int72_stk
22683 000000D1 00000000                 old72:      dd      0
22684 000000D5 4B42                         dw      424Bh
22685                                   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
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
0000021B B87000                     mov     ax,DOSBIODATASEG
0000021E 8ED8                       mov     ds,ax

                                    ;test   byte [08D0h],1          ; (MSDOS 6.21, IO.SYS - SYSINIT:021Eh)
00000220 F606[1208]01               test    byte [IsWin386],1 ; (retrodos4.sys, offset: ****h)
00000225 1F                         pop     ds
00000226 58                         pop     ax
00000227 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
00000229 9A[1308]7000               call    DOSBIODATASEG:V86_Crit_SetFocus ; 0070h:08D1h
                                ;
                                ;SR; We do not bother about the returned status of this call.
                                ;
                                fatal_loop:
0000022E AC                         lodsb
0000022F 3C24                       cmp     al,'$'
00000231 7408                       je      short fatal_done

00000233 B307                       mov     bl,7
00000235 B40E                       mov     ah,14
00000237 CD10                       int     10h                     ; whoops, this enables ints
00000239 EBF3                       jmp     short fatal_loop

                                fatal_done:
0000023B 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

```

```
23055                                   ;DOSSIZE: ; Retro DOS 2.0 feature - 25/02/2018
23056                                   ;   dw      0   ; 'MSDOS.BIN' kernel size in words
23057                                   
23058                                   ; 22/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
23059                                   ; (MSDOS 5.0 MSDOS.SYS size is 37394 bytes)
23060                                   ;DOSSIZE equ       0A000h  ; (MSDOS 6.0 - SYSINIT1.ASM - 1991)
23061                                   ; 30/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
23062                                   ; 03/09/2023 (PCDOS 7.1 IBMDOS.COM size is 42566 bytes, 04/12/2003)
23063                                   DOSSIZE equ 0B000h  ; (PCDOS 7.1 - SYSINIT)
23064                                   
23065                                   DEVICE_LIST:
23066 00000275 00000000                       dd      0
23067                                   
23068                                   ; 04/06/2018 - Retro DOS v3.0
23069                                   ; 28/03/2018
23070                                   ;; MSDOS 3.3 - SYSINIT1.ASM - 24/07/1987
23071                                   ;
23072                                   sysi_country:
23073 00000279 00000000                       dd      0 ; 5/29/86 Pointer to country table in DOS
23074                                   
23075                                   ; MSDOS 6.0
23076 0000027D 00000000                 dos_segreinit:     dw      0,0     ; room for dword
23077                                   
23078 00000281 0000                     lo_doscod_size:    dw      0       ; dos code size when in low mem
23079 00000283 0000                     hi_doscod_size:    dw      0       ; dos code size when in HMA
23080                                   
23081 00000285 0000                     def_php:    dw      0
23082                                   
23083                                   ; M022--
23084                                   ; pointer for calling into Bios_Code for re-initializing segment values.
23085                                   ;  call with ax = new segment for Bios_Code. Notice that we'll
23086                                   ;  call it in its temporary home, cuz seg_reinit won't get moved to
23087                                   ;  the new home.
23088                                   
23089                                   ;Bios_Code  equ    KERNEL_SEGMENT  ; 0070h
23090                                   ; 21/10/2022
23091                                   ;DOSBIOCODESEG     equ     02C7h ; (MSDOS 5.0 IO.SYS)
23092                                   
23093                                   ; 22/10/2022
23094                                   seg_reinit_ptr:    ; label dword
23095 00000287 [3200]                             dw      seg_reinit ; Bios_Code:0032h for MSDOS 6.21 IO.SYS
23096                                   temp_bcode_seg:
23097                                             ;dw     Bios_Code  ; 02CCh for MSDOS 6.21 IO.SYS
23098                                             ; 22/10/2022
23099 00000289 2D03                             dw      DOSBIOCODESEG ; 02C7h for MSDOS 5.0 IO.SYS
23100                                                             ; 364h for PCDOS 7.1 IBMBIO.COM - 29/12/2023
23101                                   fake_floppy_drv:
23102 0000028B 00                               db      0       ; set to 1 if this machine
23103                                                           ; does not have any floppies!!!
23104                                   
23105                                   ; Internal Stack Parameters
23106                                   
23107 0000028C 0900                     stack_count:       dW      defaultcount ; 9
23108 0000028E 8000                     stack_size: dw     defaultsize  ; 128
23109 00000290 00000000                 stack_addr: dd     0
23110                                   
23111                                   ; 05/06/2018 - Retro DOS v3.0
23112                                   
23113                                   ; various default values
23114                                   
23115 00000294 0100                     MEMORY_SIZE:       dw      1
23116                                   
23117                                   ; 22/03/2019 - Retro DOS v4.0 (MSDOS 6.0 source, MSDOS 6.21 disassembled src.)
23118                                   
23119 00000296 0000                     RPLMemTop:  dw     0  ; 22/10/2022 (MSDOS 5.0 IO.SYS SYSINIT:0294h)
23120 00000298 00                       DEFAULT_DRIVE:     db      0       ; initialized by ibminit.
23121 00000299 FFFF                     buffers:    dw     0FFFFh  ; initialized during buffer allocation
23122 0000029B 0000                     h_buffers:  dw     0       ; # of the heuristic buffers. initially 0.
23123 0000029D 0000                     singlebuffersize: dw      0       ; maximum sector size + buffer head
23124                                   
23125 0000029F 08                       FILES:      db     8       ; enough files for pipe
23126 000002A0 04                       FCBS:       db     4       ; performance for recycling
23127 000002A1 00                       KEEP:       db     0       ; keep original set
23128 000002A2 05                       NUM_CDS: db 5       ; 5 net drives
23129                                   
23130                                   ; 22/10/2022 (MSDOS 5.0 SYSINIT)
23131                                   ;;CONFBOT: dw      0
23132                                   ;;ALLOCLIM: dw     0
23133                                   ;CONFBOT: ; 02/11/2022
23134                                   ;top_of_cdss: dw 0
23135                                   
23136                                   ; 30/12/2022 - Retrodos v4.2 (MSDOS 6.21 SYSINIT)
23137                                   ; (SYSINIT:02A3h)
23138 000002A3 0000                     CONFBOT: dw 0
23139 000002A5 0000                     ALLOCLIM: dw       0
23140 000002A7 0000                     top_of_cdss: dw 0
23141                                   
23142                                   ; 02/11/2022 (MSDOS 5.0 SYSINIT)
23143                                   ; 30/12/2022 (MSDOS 6.21 SYSINIT)
23144                                   ;ALLOCLIM: dw      0       ; (SYSINIT:02A3h)
23145                                   
23146 000002A9 413A5C00                 DirStrng: db       "A:\",0 ; string for the root directory of a drive
23147                                   
23148                                   ; 30/12/2022 - Retro DOS v4.2 (MSDOS 6.21 SYSINIT)
23149                                   %if 0
23150                                   ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
23151                                   ; (SYSINIT:02A9h)
23152                                   
23153                                   command_line:
23154                                       db      2,0
23155                                       db      'P'
23156                                       db      0
23157                                       times   124 db 0 ; db 124 dup(0)
23158                                   
23159                                   %endif
23160                                   
23161                                   ; (SYSINIT:0329h)
23162 000002AD 00                       ZERO:       db     0
23163 000002AE 00                       sepchr:     db     0
23164 000002AF 0000                     linecount: dw      0                       ; line count in config.sys
23165 000002B1 20202020200D0A24         showcount: db      '     ',cr,lf,'$'       ; used to convert linecount to ascii.
23166 000002B9 0000                     buffer_linenum: dw 0               ; line count for "buffers=" command if entered.
23167                                   
23168 000002BB FF                       sys_model_byte:    db      0FFh            ; model byte used in sysinit
23169 000002BC 00                       sys_scnd_model_byte: db 0          ; secondary model byte used in sysinit
23170                                   
23171 000002BD 0000                     buf_prev_off:      dw      0
23172                                   
23173                                           ;IF      NOT NOEXEC
23174                                   ;COMEXE EXEC0 <0,COMMAND_LINE,DEFAULT_DRIVE,ZERO>
23175                                           ;ENDIF
23176                                   
23177                                   ; 29/12/2023
23178                                   ; 01/05/2018
```

```
23179                                   COMEXE:
23180 000002BF 0000                     EXEC0.ENVIRON:      dw      0       ; seg addr of environment
23181 000002C1 [BB4B]                   EXEC0.COM_LINE:     dw      command_line ; pointer to asciz command line
23182 000002C3 0000                                 dw      0       ; SYSINIT segment (0473h for MSDOS 6.21 IO.SYS)
23183                                                               ; SYSINIT segment (0544h for PCDOS 7.1 IBMBIO.COM)
23184 000002C5 [9802]                   EXEC0.5C_FCB:       dw      DEFAULT_DRIVE ; default fcb at 5C
23185 000002C7 0000                                 dw      0       ; SYSINIT segment (0473h for MSDOS 6.21 IO.SYS)
23186 000002C9 [AD02]                   EXEC0.6C_FCB:       dw      ZERO    ; default fcb at 6C
23187 000002CB 0000                                 dw      0
23188
23189                                   ; variables for install= command.
23190
23191 000002CD 00                       multi_pass_id:      db      0               ; parameter passed to multi_pass
23192                                                                       ;  indicating the pass number
23193                                                                       ; 0 - do scan for DOS=HIGH/LOW
23194                                                                       ; 1 - load device drivers
23195                                                                       ; 2 - was to load IFS
23196                                                                       ;       now it is unused
23197                                                                       ; 3 - do install=
23198                                                                       ; >3 - nop
23199 000002CE 0000                     install_flag:       dw      0
23200
23201                                   have_install_cmd equ      00000001b       ; config.sys has install= commands
23202                                   has_installed       equ     00000010b       ; sysinit_base installed.
23203
23204 000002D0 0000                     config_size:        dw      0               ; size of config.sys file. set by sysconf.asm
23205 000002D2 00000000                 sysinit_base_ptr:   dd      0               ; pointer to sysinit_base
23206 000002D6 00000000                 sysinit_ptr:        dd      0               ; returning addr. from sysinit_base
23207 000002DA 0000                     checksum:   dw      0               ; used by sum_up
23208
23209 000002DC 20<rep 14h>              ldexec_fcb: times 20 db 20h ; db 20 dup (' ') ;big enough
23210 000002F0 00                       ldexec_line:        db      0               ;# of parm characters
23211 000002F1 20                       ldexec_start:       db      ' '
23212 000002F2 00<rep 50h>              ldexec_parm:        times 80 db 0  ; db 80 dup (0)
23213
23214                                   ;instexe exec0      <0,ldexec_line,ldexec_fcb,ldexec_fcb>
23215
23216                                   instexe:
23217 00000342 0000                     iexec.environ:      dw      0               ; seg addr of environment
23218 00000344 [F002]                   iexec.ldexec_line:  dw      ldexec_line ; pointer to asciz command line
23219 00000346 0000                                 dw      0       ; SYSINIT segment (0473h for MSDOS 6.21 IO.SYS)
23220                                                               ; SYSINIT segment (0544h for PCDOS 7.1 IBMBIO.COM)
23221 00000348 [DC02]                   iexec.ldexec_5c_fcb: dw     ldexec_fcb      ; default fcb at 5C
23222 0000034A 0000                                 dw      0       ; SYSINIT segment (0473h for MSDOS 6.22 IO.SYS)
23223 0000034C [DC02]                   iexec.ldexec_6c_fcb: dw     ldexec_fcb      ; default fcb at 6C
23224 0000034E 0000                                 dw      0
23225
23226                                   ; variables for comment=
23227
23228 00000350 00                       com_level:  db      0               ; level of " " in command line
23229 00000351 00                       cmmt:               db      0               ; length of comment string token
23230 00000352 00                       cmmt1:              db      0               ; token
23231 00000353 00                       cmmt2:              db      0               ; token
23232 00000354 00                       cmd_indicator:      db      0
23233 00000355 00                       donotshownum:       db      0
23234
23235 00000356 0000                     count:              dw      0
23236 00000358 0000                     org_count:  dw      0
23237 0000035A 0000                     chrptr:             dw      0
23238 0000035C 0000                     cntryfilehandle: dw         0
23239 0000035E 0000                     old_area:   dw      0
23240 00000360 0000                     impossible_owner_size: dw 0             ; paragraph
23241
23242                                   bucketptr: ; label dword
23243                                   bufptr:         ; label dword           ; leave this stuff in order!
23244 00000362 0000                     memlo:      dw      0
23245                                   prmblk:         ; label word
23246 00000364 0000                     memhi:      dw      0
23247 00000366 0000                     ldoff:      dw      0
23248 00000368 0000                     area:       dw      0
23249
23250                                   ; 29/12/2023 - PCDOS 7.1 IBMBIO.COM - SYSINIT:036Ah
23251 0000036A 0000                     prev_memhi: dw 0
23252 0000036C 0000                     prev_alloclim:      dw 0
23253 0000036E 00                       dosdata_umb:        db 0
23254
23255                                   ; Following is the request packet used to call INIT routines for
23256                                   ; all device drivers. Some fields may be accessed individually in
23257                                   ; the code, and hence have individual labels, but they should not
23258                                   ; be separated.
23259
23260 0000036F 19                       packet:     db      25                      ; PCDOS 7.1 IBMBIO.COM
23261                                       ;db     24                      ; was 22
23262 00000370 00                           db      0
23263 00000371 00                           db      0                       ; initialize code
23264 00000372 0000                         dw      0
23265 00000374 00<rep 8h>                   times   8 db 0  ; db 8 dup (?)
23266
23267 0000037C 00                       unitcount:  db      0
23268 0000037D 00000000                 break_addr: dd      0
23269 00000381 00000000                 bpb_addr:   dd      0
23270                                   drivenumber:        ; 22/10/2022
23271 00000385 00                       devdrivenum:        db      0
23272 00000386 0000                     configmsgflag:      dw      0  ; used to control "error in config.sys line #" message
23273
23274                                   ; end of request packet
23275
23276                                   ;drivenumber:       db      0  ; 22/03/2019
23277
23278                                   toomanydrivesflag:
23279 00000388 00                                   db      0  ; >24 fixed disk partitions flag ; M029
23280 00000389 90                       align 2
23281
23282                                   BCodeSeg:   ; 21/10/2022
23283 0000038A 2D03                         dw      DOSBIOCODESEG ; (02C7h for MSDOS 5.0 IO.SYS)
23284                                                       ; 0364h for PCDOS 7.1 IBMBIO.COM - 29/12/2023
23285                                       ;dw     Bios_Code  ; = KERNEL_SEGMENT = 0070h (for Retro DOS v4.0)
23286                                                         ; BCodeSeg = 2CCh (for MSDOS 6.21 IO.SYS)
23287
23288                                   ; 30/12/2022
23289                                   ; MSDOS 6.21 IO.SYS, SYSINIT:0387h
23290                                   ;
23291                                   ; Magicbackdoor: dd 0
23292                                   ; NullBackdoor:
23293                                   ;           retf
23294
23295                                   ; 02/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
23296                                   ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
23297                                   ; 19/04/2019
23298                                   _timer_lw_:
23299 0000038C 0000                         dw      0  ; MSDOS 6.21 IO.SYS - SYSINIT:038Ch
23300
23301                                   ; 29/12/2023 - Retro DOS v5.0
23302                                   ; PCDOS 7.1 IBMBIO.COM - SYSINIT:038Eh
```

```
23303
23304 0000038E 00                       F5_key:     db 0
23305 0000038F 00                       F8_key:     db 0
23306 00000390 00000000                 MagicBackdoor:     dd 0
23307                                   NullBackdoor:
23308 00000394 CB                                   retf
23309
23310                                   ;SR;
23311                                   ; This is the communication block between the DOS and the BIOS. It starts at
23312                                   ;the SysinitPresent flag. Any other data that needs to be communicated
23313                                   ;to the DOS should be added after SysinitPresent. The pointer to this block
23314                                   ;is passed to DOS as part of the DOSINIT call.
23315                                   ;
23316
23317                                   BiosComBlock:
23318                                       ;dd    Bios_Data:SysinitPresent
23319                                              ; 0070h:08FDh for MSDOS 6.21 IO.SYS
23320 00000395 [DD07]                     dw     SysinitPresent  ; (retrodos4.sys, offset: ****h)
23321                                       ;dw    KERNEL_SEGMENT ; 0070h
23322                                       ; 21/10/2022
23323 00000397 7000                       dw     DOSBIODATASEG ; 0070h
23324
23325                                   ;align 2
23326
23327                                       ; 22/10/2022 - (MSDOS 5.0 IO.SYS, SYSINIT:0406h)
23328                                       ; 30/12/2022 - (MSDOS 6.21 IO.SYS, SYSINIT:0392h)
23329                                   tempstack:
23330 00000399 00<rep 80h>                 times  128 db 0  ; db 80h dup (?)
23331
23332                                   ; ---------------------------------------------------------------------------
23333
23334                                       ; 29/12/2023 - Retro DOS v5.0
23335                                       ; 22/10/2022 - Retro DOS v4.0
23336                                       ;      ; (MSDOS 5.0 IO.SYS, SYSINIT:0486h)
23337                                   GOINIT:            ; (MSDOS 6.22 IO.SYS, SYSINIT:0412h)
23338                                              ; (PCDOS 7.1 IBMBIO.COM, SYSINIT:0419h)
23339                                       ; 12/12/2023
23340 00000419 0E                          push    cs
23341 0000041A 1F                          pop     ds
23342
23343                                       ; 12/12/2022
23344                                       ; 22/03/2019 - Retro DOS v4.0
23345                                       ; 06/07/2018
23346                                       ; 04/06/2018 - Retro DOS v3.0
23347                                   ; before doing anything else, let's set the model byte
23348 0000041B B4C0                        mov     ah,0C0h                ; get system configuration
23349 0000041D CD15                        int     15h                    ;
23350 0000041F 7214                        jc      short no_rom_config
23351
23352                                      ;cmp    ah,0                   ; double check
23353                                      ;jne    short no_rom_config
23354                                      ; 03/09/2023
23355 00000421 08E4                        or      ah,ah
23356 00000423 7510                        jnz     short no_rom_config
23357
23358                                      ; 12/12/2023 ; *
23359                                      ; ds = cs
23360
23361 00000425 268A4702                    mov     al,[es:bx+ROMBIOS_DESC.bios_sd_modelbyte]
23362                                      ;mov    [cs:sys_model_byte],al
23363 00000429 A2[BB02]                    mov     [sys_model_byte],al ; *
23364 0000042C 268A4703                    mov     al,[es:bx+ROMBIOS_DESC.bios_sd_scnd_modelbyte]
23365                                      ;mov    [cs:sys_scnd_model_byte],al
23366 00000430 A2[BC02]                    mov     [sys_scnd_model_byte],al ; *
23367                                      ;jmp    short SYSIN
23368                                      ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
23369 00000433 EB29                        jmp     short move_myself
23370
23371                                   no_rom_config:                          ; Old ROM
23372                                      ; 12/12/2023
23373                                      ;mov    ax,0F000h
23374                                      ;mov    ds,ax
23375                                      ;mov    al,[0FFFEh]
23376                                      ;mov    [cs:sys_model_byte],al; set the model byte.
23377                                      ; 12/12/2023
23378                                      ; ds = cs
23379 00000435 B800F0                     mov     ax,0F000h
23380 00000438 8EC0                       mov     es,ax
23381 0000043A 26A0FEFF                   mov     al,[es:0FFFEh]
23382 0000043E A2[BB02]                   mov     [sys_model_byte],al   ; set the model byte.
23383
23384                                   ; set fake_floppy_drv if there is no diskette drives in this machine.
23385                                   ; execute the equipment determination interrupt and then
23386                                   ; check the returned value to see if we have any floppy drives
23387                                   ; if we have no floppy drive we set cs:fake_floppy_drv to 1
23388                                   ; see the at tech ref bios listings for help on the equipment
23389                                   ; flag interrupt (11h)
23390
23391                                      ; 22/10/2022
23392                                   ;check_for_fake_floppy:                 ; entry point for rom_config above
23393 00000441 CD11                        int     11h                    ; check equipment flag
23394
23395                                      ; 29/12/2023 - Retro DOS v5.0
23396                                      ;jmp    short check_for_fake_floppy
23397                                      ; PCDOS 7.1 IBMBIO.COM - SYSINIT:0446h
23398                                      ;db     52h     ; 'RPS' sign
23399                                      ;db     50h
23400                                      ;db     53h
23401
23402                                   check_for_fake_floppy:
23403                                      ; 29/12/2023
23404                                      ; PCDOS 7.1 IBMBIO.COM - SYSINIT:0449h
23405                                      ;or     ax, 1          ; (nonsense! this may be overwritten/disabled
23406                                      ;                      ; by using 'RPS' sign position)
23407                                      ;                      ;              03/07/2023 - Erdogan Tan
23408                                      ;test   ax, 1          ; have any floppies?
23409
23410                                      ; 12/12/2022
23411 00000443 A801                       test    al,1
23412                                      ;test   ax,1                   ; have any floppies?
23413 00000445 7517                       jnz     short move_myself      ; yes,normal system
23414
23415                                   ; Some ROM BIOSs lie that there are no floppy drives. Lets find out
23416                                   ; whether it is an old ROM BIOS or a new one
23417                                   ;
23418                                   ; WARNING !!!
23419                                   ;
23420                                   ; This sequence of code is present in MSINIT.ASM also. Any modification
23421                                   ; here will require an equivalent modification in MSINIT.ASM also
23422
23423                                      ; 12/12/2023
23424                                      ;push   es  ; not necessary
23425
23426 00000447 30C9                       xor     cl,cl
```

```
                                    mov     ah,8                    ; get disk parameters
                                    mov     dl,0                    ; of drive 0
0000044D CD13                       int     13h
                                    ;pop    es  ; 12/12/2023
0000044F 720D                       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
00000451 08C9                       or      cl,cl
00000453 7409                       jz      short move_myself
00000455 08D2                       or      dl,dl                   ; number of flp drvs == 0?
00000457 7505                       jnz     short move_myself       ; no
                                    ;mov    byte [cs:fake_floppy_drv],1 ; set fake flag.
                                    ; 12/12/2023
                                    ; ds = cs
00000459 C606[8B02]01               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]
0000045E 8B0E[9402]                 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
00000462 49                         dec     cx
                                    ;------ Check if an RPL program is present at TOM and do not tromp over it
00000463 31DB                       xor     bx,bx
00000465 8EC3                       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
00000467 26C41EBC00                 les     bx,[es:(2Fh*4)]
0000046C 26817F035250               cmp     word [es:bx+3],'RP'
00000472 751B                       jne     short NoRPL
00000474 26807F054C                 cmp     byte [es:bx+5],'L'
00000479 7514                       jne     short NoRPL
0000047B 89CA                       mov     dx,cx                   ; get TOM into DX
0000047D 52                         push    dx
0000047E B8064A                     mov     ax,4A06h
                                    ;mov    ax,(multMULT<<8)+multMULTRPLTOM
00000481 CD2F                       int     2Fh                     ; Get new TOM from any RPL
00000483 58                         pop     ax
00000484 89D1                       mov     cx,dx
00000486 39C2                       cmp     dx,ax
00000488 7405                       je      short NoRPL
                                    ; 11/12/2022
                                    ; ds = cs
0000048A 8916[9602]                 mov     [RPLMemTop],dx
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;mov    [cs:RPLMemTop],dx
0000048E 49                         dec     cx
                                NoRPL:
0000048F B8[1054]                   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)
00000492 E80509                     call    off_to_para
```

```
00000495 29C1                       sub     cx,ax

                                ; we need to leave room for the DOS and (if not ROMDOS) for the BIOS
                                ; code above sysinit in memory
                                ;
00000497 81E9000B                   sub     cx,DOSSIZE/16 ; (0A00h)       ; leave this much room for DOS
                                                          ; (0B00h) ; (PCDOS 7.1 IBMBIO.COM) -03/09/2023-

0000049B B8701D                     mov     ax,BCODE_END            ; (1A60h for MSDOS 5.0 IO.SYS)
                                                                    ; (1A70h for MSDOS 6.21 IO.SYS)
                                                                    ; 03/09/2023
                                                                    ; (1E00h for PCDOS 7.1 IBMBIO.COM)
0000049E E8F908                     call    off_to_para             ; leave this much room for BIOS code
000004A1 29C1                       sub     cx,ax
000004A3 8EC1                       mov     es,cx                   ; segment where sysinit will be located

                                    ; 12/12/2023
000004A5 FC                         cld     ; not necessary               ; set up move
000004A6 31F6                       xor     si,si
000004A8 89F7                       mov     di,si

000004AA B9[1054]                   mov     cx,SI_end               ; (sysinit code size)
000004AD D1E9                       shr     cx,1                    ; divide by 2 to get words
000004AF F3A5                       rep     movsw                   ; relocate sysinit

000004B1 06                         push    es                      ; push relocated segment
000004B2 B8[B704]                   mov     ax,SYSIN
000004B5 50                         push    ax                      ; push relocated entry point

000004B6 CB                         retf                            ; far jump to relocated sysinit

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

                                ; 30/12/2023
                                ; PCDOS 7.1 IBMBIO.COM - SYSINIT:04CEh
                                %if 0
                                get_cpu_type:
                                    pushf
                                    push    bx
                                    xor     bx,bx
                                    xor     ax,ax
                                    push    ax
                                    popf
                                    pushf
                                    pop     ax
                                    and     ax,0F000h
                                    cmp     ax,0F000h
                                    jz      short cpu_8086
                                    mov     ax,0F000h
                                    push    ax
                                    popf
                                    pushf
                                    pop     ax
                                    and     ax,0F000h
                                    jz      short cpu_286
                                cpu_386:
                                    inc     bx
                                cpu_286:
                                    inc     bx
                                cpu_8086:
                                    mov     ax,bx
                                    pop     bx
                                    popf
                                    retn
                                %endif

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

                                ;   MOVE THE DOS TO ITS PROPER LOCATION

                                    ; 22/10/2022 - Retro DOS 4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:0533h)
                                    ; 30/12/2022 - Retro DOS 4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:04BFh)
                                    ; 03/09/2023 - Retro DOS 4.2 (5.0 - Modified PCDOS 7.1 IBMBIO.COM)
                                    ; (SYSINIT:04F3h)
                                SYSIN:
                                    ; Retro DOS 5.0 - 30/12/2023
                                    ; Retro DOS 4.0 - 22/03/2019
                                    ; Retro DOS 2.0 - 25/02/2018

                                    ; 23/04/2019
                                    ;;mov   ax,Bios_Data
                                    ;mov    ax,KERNEL_SEGMENT ; 0070h
                                    ; 21/10/2022
000004B7 B87000                     mov     ax,DOSBIODATASEG ; 0070h
000004BA 8ED8                       mov     ds,ax

                                    ; 30/12/2023 - Retro DOS v5.0
                                    ;;;
                                    ;push   es
                                    ;push   ax              ; not needed (*) E.TAN - 03/07/2023
                                    ;push   di

                                    ;call   get_cpu_type    ; determine if 386 system
                                    ;
                                get_cpu_type:
000004BC 9C                         pushf
000004BD 31C0                       xor     ax,ax
000004BF 50                         push    ax
000004C0 9D                         popf
000004C1 9C                         pushf
000004C2 58                         pop     ax
000004C3 2500F0                     and     ax,0F000h
000004C6 3D00F0                     cmp     ax,0F000h
000004C9 740F                       jz      short cpu_8086
000004CB B800F0                     mov     ax,0F000h
000004CE 50                         push    ax
000004CF 9D                         popf
000004D0 9C                         pushf
000004D1 58                         pop     ax
000004D2 2500F0                     and     ax,0F000h
000004D5 7402                       jz      short cpu_286
                                cpu_386:
000004D7 29C0                       sub     ax,ax
                                cpu_286:
000004D9 40                         inc     ax
                                cpu_8086:   ; ax = 0
                                    ; 30/12/2023 - Retro DOS v5.0
000004DA 2EA2[B606]                 mov     [cs:cpu_type],al ; 07/04/2024
000004DE 9D                         popf
                                    ;
                                    ;cmp    ax,2            ; 0 = 8086, 1 = 286, 2 = 386
000004DF 3C02                       cmp     al,2
```

```
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,2048D  ; 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
00000613 A3[9D02]                   mov     [singlebuffersize],ax ; total size for a buffer
                                    ;mov    [cs:singlebuffersize],ax
                                    ; 11/12/2022
00000616 A0[9802]                   mov     al,[DEFAULT_DRIVE]    ; get the 1 based boot drive number set by msinit
                                    ;mov    al,[cs:DEFAULT_DRIVE]
                                    ;mov    [es:di+SYSI_BOOT_DRIVE],al ; set sysi_boot_drive
00000619 26884543                   mov     [es:di+43h],al

                                ; determine if 386 system...

                                ; 30/12/2023
                                %if 0
                                    ;get_cpu_type               ; macro to determine cpu type

                                get_cpu_type:
                                    ; 11/12/2022
                                    pushf
                                    ;push   bx
                                    ;xor    bx,bx
                                    ; 11/12/2022
                                    ;xor    cx,cx
                                    ;
                                    xor     ax,ax
                                    ; ax = 0
                                    push    ax
                                    popf
                                    pushf
                                    pop     ax
                                    and     ax,0F000h
                                    ;cmp    ax,0F000h
                                    cmp     ah,0F0h
                                    je      short cpu_8086
                                    ;mov    ax,0F000h
                                    mov     ah,0F0h
                                    ; ax = 0F000h
                                    push    ax
                                    popf
                                    pushf
                                    pop     ax
                                    ;and    ax,0F000h
                                    and     ah,0F0h
                                    jz      short cpu_286
                                cpu_386:
                                    ; 11/12/2022
                                    ;;inc   bx
                                    ;inc    cx
                                    ; 11/12/2022
                                    ;mov    byte [es:di+SYSI_DWMOVE],1
                                    mov     byte [es:di+44h],1

                                    ; 03/09/2023 - Retro DOS v5.0 (PCDOS 7.1 Modified SYSINIT)
                                    ; change A20 line on/off check code to the faster (for 32 bit cpu)
                                    ;push   es
                                    ;push   di
                                    ;mov    ax,DOSBIODATASEG ; 0070h
                                    ;mov    es,ax
                                    ;cld
                                    ;mov    di,cpu386_cmpsd ; (IsA20Off)
                                    ;mov    ax,4B9h         ; mov cx,4 ; B90400
                                    ;stosw
                                    ;mov    ax,0F300h       ; repz  ; F3
                                    ;stosw
                                    ;mov    ax,0A766h       ; cmpsd ; 66A7
                                    ;stosw
                                    ;pop    di
                                    ;pop    es

                                cpu_286:
                                    ;;;inc  bx
                                    ;;inc   cx
                                cpu_8086:
                                    ; 11/12/2022
                                    ;;mov   ax,bx
                                    ;pop    bx
                                    popf
                                %endif
                                    ;...

                                    ; 11/12/2022
                                    ;or     cl,cl
                                    ;jz     short not_386_system
                                    ; 11/12/202
                                    ;cmp    cl,2
                                    ;;cmp   ax,2                    ; is it a 386?
                                    ;jne    short not_386_system   ; no: don't mess with flag

                                    ; 30/12/2023 - Retro DOS v5.0
0000061D 803E[B606]02               cmp     byte [cpu_type], 2     ; is it a 386?
00000622 7505                       jne     short _not_386_cpu     ; no: don't mess with flag

                                    ;mov    byte [es:di+SYSI_DWMOVE],1
                                    ; 11/12/2022
                                    ; 22/10/2022
00000624 26C6454401                 mov     byte [es:di+44h],1
                                _not_386_cpu:
                                    ;mov    al,[es:di+SYSI_NUMIO]
00000629 268A4520                   mov     al,[es:di+20h]
                                    ; 11/12/2022
                                    ; ds = cs
0000062D A2[8503]                   mov     [drivenumber],al       ; save start of installable block drvs
                                    ;mov    [cs:drivenumber],al

00000630 8CC8                       mov     ax,cs
00000632 83E811                     sub     ax,11h                 ; room for PSP we will copy shortly
                                    ; 11/12/2022
                                    ;mov    cx,[singlebuffersize] ; temporary single buffer area
                                    ;;mov   cx,[cs:singlebuffersize]
                                    ;shr    cx,1
                                    ;shr    cx,1                   ; divide size by 16...
                                    ;shr    cx,1
                                    ;shr    cx,1                   ; ...to get paragraphs...
                                    ;inc    cx                     ; ... and round up
                                    ; 11/12/2022
00000635 8B1E[9D02]                 mov     bx,[singlebuffersize]
00000639 B104                       mov     cl,4
0000063B D3EB                       shr     bx,cl
0000063D 43                         inc     bx

                                ;   cas note: this unorthodox paragraph rounding scheme wastes a byte
                                ;     if [singlebuffersize] ever happens to be zero mod 16. Could this
                                ;     ever happen? Only if the buffer overhead was zero mod 16, since
                                ;     it is probably safe to assume that the sector size always will be.
                                ;
```

```
24171                                   ;    mohans also found a bug in CONFIG.SYS processing where it replaces
24172                                   ;     EOF's with cr,lf's, without checking for collision with [confbot].
24173                                   ;     perhaps the extra byte this code guarantees is what has kept that
24174                                   ;     other code from ever causing a problem???
24175
24176                                       ; 11/12/2022
24177 0000063E 29D8                       sub     ax,bx
24178                                     ;sub    ax,cx
24179 00000640 A3[A702]                   mov     [top_of_cdss],ax      ; temp "unsafe" location
24180                                     ; 22/10/2022
24181                                     ;mov    [cs:top_of_cdss],ax
24182
24183                                   ;   chuckst -- 25 Jul 92 -- added code here to pre-allocate space
24184                                   ;   for 26 temporary CDSs, which makes it easier to use alloclim
24185                                   ;   for allocating memory for MagicDrv.
24186
24187                                       ; 30/12/2023
24188                                     ;push  es ; not necessary (!*)      ; preserve pointer to dosinfo
24189                                     ;push  di
24190
24191                                       ; 22/10/2022
24192                                   ;   mov     cx,ax                 ; save pointer for buffer
24193                                   ;
24194                                   ;;  now allocate space for 26 CDSs
24195                                   ;
24196                                   ;   sub     ax,((26 *(curdirlen))+15)/16
24197                                   ;   mov     [ALLOCLIM],ax         ; init top of free memory pointer
24198                                   ;   mov     [CONFBOT],ax          ; init this in case no CONFIG.SYS
24199
24200                                       ; 30/12/2022 - Retro DOS v4.2 (MSDOS 6.21 IO.SYS)
24201                                       ; (SYSINIT:064Ch)
24202 00000643 89C1                       mov     cx,ax ; (*)
24203 00000645 2D8F00                     sub     ax,((26*(curdirlen))+15)/16 ; sub ax,143
24204 00000648 A3[A502]                   mov     [ALLOCLIM],ax         ; init top of free memory pointer
24205 0000064B A3[A302]                   mov     [CONFBOT],ax          ; init this in case no CONFIG.SYS
24206
24207                                   ; setup and initialize the temporary buffer at cx
24208
24209                                     ;les    di,[es:di+SYSI_BUF]   ; get the buffer chain entry pointer
24210 0000064E 26C47D12                   les     di,[es:di+12h]
24211                                     ; 11/12/2022
24212 00000652 31DB                       xor     bx,bx
24213                                     ;xor    ax,ax
24214                                     ;mov    [es:di+BUFFINF.Dirty_Buff_Count],ax ; 0
24215                                     ;mov    word [es:di+4],0
24216 00000654 26895D04                   mov     [es:di+4],bx ; 0
24217                                     ;mov    [es:di+BUFFINF.Buff_Queue],ax ; 0
24218                                     ;mov    word [es:di],0
24219 00000658 26891D                     mov     [es:di],bx ; 0
24220                                     ;mov    [es:di+BUFFINF.Buff_Queue+2],cx ; cx = [top_of_cdss] ; 6.21
24221                                     ;;mov   [es:di+BUFFINF.Buff_Queue+2],ax ; ax = [top_of_cdss] ; 5.0
24222                                     ;mov    [es:di+2],ax
24223                                     ;mov    es,ax  ; [top_of_cdss] = [CONFBOT]
24224                                     ; 30/12/2022 - Retro DOS v4.2 (MSDOS 6.21 IO.SYS, SYSINIT)
24225 0000065B 26894D02                   mov     [es:di+2],cx ; [top_of_cdss] ; (*)
24226 0000065F 8EC1                       mov     es,cx
24227
24228                                     ; 11/12/2022
24229                                     ;xor    ax,ax
24230                                     ;mov    di,ax                 ; es:di -> single buffer
24231 00000661 89DF                       mov     di,bx
24232                                     ; di = 0
24233
24234                                     ;mov    [es:di+buffinfo.buf_next],ax ; points to itself
24235                                     ; 11/12/2022
24236                                     ;mov    [es:di],ax ; 0
24237 00000663 26891D                     mov     [es:di],bx ; 0
24238                                     ;mov    [es:di+buffinfo.buf_prev],ax ; points to itself
24239                                     ; 11/12/2022
24240                                     ;mov    [es:di+2],ax ; 0
24241 00000666 26895D02                   mov     [es:di+2],bx ; 0
24242
24243                                     ; 22/10/2022 - Retro DOS 4.0 (Modified MSDOS 5.0 IO.SYS SYINIT)
24244                                     ; MSDOS 5.0 IO.SYS - SYSINIT:06E0h
24245
24246                                     ;mov    word [es:di+buffinfo.buf_ID],00FFh ; free buffer,clear flag
24247 0000066A 26C74504FF00               mov     word [es:di+4],00FFh
24248                                   ;SYSINIT:06E6h
24249                                       ;;mov   [es:di+buffinfo.buf_sector],ax ; 0
24250                                       ;mov    word [es:di+6],0
24251                                       ; 11/12/2022
24252                                       ;mov    [es:di+buffinfo.buf_sector],bx ; 0
24253 00000670 26895D06                   mov     [es:di+6],bx ; 0
24254                                       ;;mov   [es:di+buffinfo.buf_sector+2],ax ; 0
24255                                       ;mov    word [es:di+8],0
24256                                       ; 11/12/2022
24257                                       ;mov    [es:di+buffinfo.buf_sector+2],bx ; 0
24258 00000674 26895D08                   mov     [es:di+8],bx ; 0
24259
24260                                       ; 30/12/2023 (!*)
24261                                     ;pop    di                    ; restore pointer to DOSINFO data
24262                                     ;pop    es
24263
24264                                       ; 11/12/2022
24265                                       ; ds = cs
24266                                       ; 22/10/2022
24267                                     ;push   cs
24268                                     ;pop    ds
24269
24270 00000678 E82807                     call    TempCDS               ; set up cdss so re_init and sysinit
24271                                                                     ;  can make disk system calls
24272                                                                     ; tempcds trashes ds
24273                                       ; 10/05/2019
24274 0000067B 2E8E1E[8502]               mov     ds,[cs:def_php]               ; retrieve pointer to PSP returned by DOSINIT
24275
24276                                     ;if not ibmjapver
24277                                     ;call  far KERNEL_SEGMENT:re_init ; re-call the bios
24278                                     ;endif
24279
24280                                       ; 22/10/2022
24281                                   ;SYSINIT:06FEh:    ; (MSDOS 5.0 IO.SYS, SYSINIT)
24282                                       ; 30/12/2022
24283                                   ;SYSINIT:0697h:    ; (MSDOS 6.21 IO.SYS, SYSINIT)
24284                                     ;call  far ptr 70h:89Bh
24285 00000680 9A[2807]7000               call    DOSBIODATASEG:RE_INIT
24286
24287 00000685 FB                         sti                           ; ints ok
24288 00000686 FC                         cld                           ; make sure
24289
24290                                   ; 23/03/2019
24291
24292                                   ;SYSINIT:069Eh     ; 30/12/2022
24293
24294                                   ; 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.

00000687 8CCB                       mov     bx,cs
00000689 83EB10                     sub     bx,10h
0000068C 8EC3                       mov     es,bx
0000068E 31F6                       xor     si,si
00000690 89F7                       mov     di,si
00000692 B98000                     mov     cx,128
00000695 F3A5                       rep     movsw

                                    ;mov    [es:PDB.JFN_POINTER+2],es ; Relocate
                                    ; 22/10/2022
00000697 268C063600                 mov     [es:36h],es

                                    ; Set Process Data Block - Program Segment Prefix address
                                    ; BX = PDB/PSP segment
0000069C B450                           mov ah,50h  ; SET_CURRENT_PDB
0000069E CD21                       int     21h                 ; tell DOS we moved it
                                                ; DOS - 2+ internal - SET PSP SEGMENT
                                                ; BX = segment address of new PSP
                                    ; 22/10/2022
                                    ; 27/03/2019
                                    ; 30/12/2023
                                    ;push   ds ; */             ; preserve DS returned by DOSINIT

000006A0 0E                         push    cs
000006A1 1F                         pop     ds

                                    ; set up temp. critical error handler
000006A2 BA[794A]                   mov     dx,int24            ; set up int 24 handler
                                    ;;mov   ax,(SET_INTERRUPT_VECTOR*256)+24h
                                    ;mov    ax,(SET_INTERRUPT_VECTOR<<8)|24h
000006A5 B82425                     mov     ax,2524h
000006A8 CD21                       int     21h

000006AA 803E[8803]00                   cmp     byte [toomanydrivesflag],0 ; Q: >24 partitions?      M029
000006AF 7406                           je      short no_err                ;  N: continue           M029
000006B1 BA[9E53]                       mov     dx,TooManyDrivesMsg         ;  Y: print error message M029
                                        ; 22/10/2022
                                        ;call   print                       ;                        M029
                                        ; 12/12/2022
000006B4 EB04                       jmp     short p_dosinit_msg ; 23/03/2019 - Retro DOS v4.0

                                    ; 30/12/2023 - Retro DOS v5.0
                                cpu_type:
000006B6 FF                         db 0FFh ; db 0

                                no_err:
                                    ; 12/05/2019
                                    ;-----------------------------------------------
                                    ; 27/06/2018 - Retro DOS v3.0 ; 23/03/2019 - Retro DOS v4.0
                                    ; 22/10/2022 - Retro DOS v4.0
                                    ; 12/12/2022
                                    ; 30/12/2023 - Retro DOS v5.0
000006B7 BA[7D4A]                   mov     dx,BOOTMES              ; Display (fake) MSDOS version message
                                p_dosinit_msg:
000006BA E89743                     call    print                   ; Print message
                                    ;-----------------------------------------------

                                    ; 11/12/2022
                                    ; 22/10/2022
                                    ; 23/03/2019 - Retro DOS v4.0
                                    ;pop    ds                      ; start of free memory
                                    ;mov    dl,[cs:DEFAULT_DRIVE]

                                    ; 11/12/2022
                                    ; 27/03/2019
000006BD 8A16[9802]                 mov     dl,[DEFAULT_DRIVE]
                                    ; 30/12/2023
                                    ;pop    ds ; */

000006C1 08D2                       or      dl,dl
                                    ; 30/12/2023
000006C3 7405                       jz      short nodrvset          ; bios didn't say
                                    ;jz     short ProcessConfig  ; (Retro DOS v4.0 does not contain DBLSPACE code)
                                    ;dec    dl                      ; A = 0
                                    ; 18/12/2022
000006C5 4A                         dec     dx
000006C6 B40E                       mov     ah,0Eh  ; SET_DEFAULT_DRIVE
000006C8 CD21                       int     21h                     ; select the disk
                                                ; DOS - SELECT DISK
                                                ; DL = new default drive number (0 = A, 1 = B, etc.)
                                                ; Return: AL = number of logical drives
                                nodrvset:
                                    ; 04/01/2023 - Retro DOS 4.2 (Modified MSDOS 6.21 IO.SYS SYINIT)
                                    ; (SYSINIT:06DFh)

                                    ; 30/12/2023 - Retro DOS 5.0
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:0733h)
000006CA 1E                         push    ds
000006CB 29C0                       sub     ax,ax
000006CD 8ED8                       mov     ds,ax ; 0 ; ROM BIOS Data Area
000006CF A16C04                     mov     ax,[46Ch] ; timer tick count (18.2 ticks per second)
                                    ;mov    [cs:_timer_lw_],ax
000006D2 1F                         pop     ds
                                    ; ds = cs
000006D3 A3[8C03]                   mov     [_timer_lw_],ax

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

                                    ;ifdef dblspace_hooks
                                    ;       ....
                                    ;       ....
                                    ;endif

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

                                    ; 30/12/2023 - Retro DOS 5.0 (Modified MSDOS 7.1 IBMBIO.COM SYS SYINIT)
                                ; ------------------------------------------------------------------------------
                                    ; PCDOS 7.1 IBMBIO.COM - SYSINIT:0740h

000006D6 0E                         push    cs
000006D7 07                         pop     es

                                    ; 07/04/2024
                                    ;mov    word [cs:MagicBackdoor+2],cs
                                    ;mov    word [cs:MagicBackdoor], NullBackdoor
000006D8 8C0E[9203]                 mov     word [MagicBackdoor+2],cs
000006DC C706[9003][9403]           mov     word [MagicBackdoor], NullBackdoor

                                    ; ds = es = cs = SYSINIT segment
                                set_drvspc_size:
000006E2 BE[AB16]                   mov     si,MagicDDName          ; "\DBLSPACE.BIN"
                                set_dblspc_size:
```

```
24419 000006E5 E8792F                   call    SizeDevice
24420 000006E8 732B                     jnc     short wait_for_key_2s
24421                                   ;cmp    byte [cs:si], 'C'
24422 000006EA 803C43                   cmp     byte [si],'C'           ; "C:\STACKER.BIN"
24423 000006ED 740C                     je      short set_drvspc_name
24424                                   ;cmp    byte [cs:DEFAULT_DRIVE],3
24425 000006EF 803E[9802]03             cmp     byte [DEFAULT_DRIVE],3
24426 000006F4 7405                     je      short set_drvspc_name
24427 000006F6 83EE02                   sub     si,2                    ; "C:\DBLSPACE.BIN"
24428 000006F9 EBEA                     jmp     short set_dblspc_size
24429
24430                                 set_drvspc_name:
24431                                   ;cmp    byte [cs:MagicDDName+2],'R'   ; "BLSPACE.BIN"
24432 000006FB 803E[AD16]52             cmp     byte [MagicDDName+2],'R'
24433 00000700 7408                     je      short set_stacker_name
24434                                   ;mov    word [cs:MagicDDName+2],'RV'  ; "DRVSPACE.BIN"
24435 00000702 C706[AD16]5256           mov     word [MagicDDName+2],'RV'
24436 00000708 EBD8                     jmp     short set_drvspc_size
24437
24438                                 set_stacker_name:
24439 0000070A 81FE[B916]               cmp     si,StackerName          ; "C:\STACKER.BIN"
24440 0000070E 734B                     jnb     short wfk2s_4
24441 00000710 BE[BB16]                 mov     si,StackerName+2        ; "\STACKER.BIN"
24442 00000713 EBD0                     jmp     short set_dblspc_size
24443
24444                                 wait_for_key_2s:
24445                                   ;mov    [cs:MagicDDNamePtr],si
24446 00000715 8936[A716]               mov     [MagicDDNamePtr],si
24447 00000719 1E                       push    ds
24448 0000071A 29C0                     sub     ax,ax
24449 0000071C 8ED8                     mov     ds,ax  ; 0              ; ROMBIOS data area
24450 0000071E 8B166C04                 mov     dx,[46Ch]               ; Counter for Interrupt 1Ah
24451                                 wfk2s_1:
24452 00000722 B401                     mov     ah,1
24453 00000724 CD16                     int     16h             ; KEYBOARD - CHECK BUFFER, DO NOT CLEAR
24454                                                           ; Return: ZF clear if character in buffer
24455                                                           ; AH = scan code, AL = character
24456                                                           ; ZF set if no character in buffer
24457 00000726 7511                     jnz     short wfk2s_2
24458 00000728 B402                     mov     ah,2
24459 0000072A CD16                     int     16h                     ; KEYBOARD - GET SHIFT STATUS
24460                                                                   ; AL = shift status bits
24461 0000072C A803                     test    al,3
24462 0000072E 7509                     jnz     short wfk2s_2
24463 00000730 A16C04                   mov     ax,[46Ch]               ; tick count
24464 00000733 29D0                     sub     ax,dx
24465 00000735 3C25                         cmp al,37                   ; 2 seconds
24466 00000737 72E9                     jb      short wfk2s_1           ; wait for user's key press
24467                                 wfk2s_2:
24468 00000739 1F                       pop     ds                      ; read/check the pressed key
24469 0000073A 29DB                     sub     bx,bx   ; bx = 0
24470 0000073C B402                     mov     ah,2
24471 0000073E CD16                     int     16h                     ; KEYBOARD - GET SHIFT STATUS
24472                                                                   ; AL = shift status bits
24473 00000740 A803                     test    al,3                    ; Left or Right SHIFT key pressed ?
24474 00000742 7402                     jz      short wfk2s_3           ; no
24475 00000744 43                       inc     bx
24476 00000745 43                       inc     bx      ; bx = 2
24477                                 wfk2s_3:
24478 00000746 B401                     mov     ah,1
24479 00000748 CD16                     int     16h             ; KEYBOARD - CHECK BUFFER, DO NOT CLEAR
24480                                                           ; Return: ZF clear if character in buffer
24481                                                           ; AH = scan code, AL = character
24482                                                           ; ZF set if no character in buffer
24483 0000074A 7418                     jz      short wfk2s_6
24484 0000074C 80FC65                   cmp     ah,65h                  ; F8 key pressed ?
24485 0000074F 740C                     jz      short wfk2s_5
24486 00000751 80FC62                   cmp     ah,62h                  ; F5 key pressed ?
24487 00000754 750E                     jnz     short wfk2s_6
24488                                   ;mov    byte [cs:F5_key],1
24489 00000756 C606[8E03]01             mov     byte [F5_key],1
24490                                 wfk2s_4:
24491 0000075B EB49                     jmp     short ProcessConfig     ; continue (as normal/default state)
24492
24493                                 wfk2s_5:
24494                                   ;mov    byte [cs:F8_key],1
24495 0000075D C606[8F03]01             mov     byte [F8_key],1
24496 00000762 EB42                     jmp     short ProcessConfig
24497
24498                                 wfk2s_6:
24499 00000764 E8AA02                   call    AllocFreeMem            ; get the largest free block from DOS
24500 00000767 E8700F                   call    MagicPreload            ; **** PRE-LOAD MAGICDRV!!! ****
24501
24502                                   ; 07/04/2024 - Retro DOS v5.0
24503                                   ; (DS may not be same with CS here!)
24504 0000076A 0E                       push    cs
24505 0000076B 1F                       pop     ds ; *
24506 0000076C 8E06[6803]               mov     es,[area]
24507
24508 00000770 09C0                     or      ax,ax                   ; error?
24509 00000772 7406                     jz      short wfk2s_7
24510                                 PreloadFailed:
24511 00000774 B449                     mov     ah,49h                  ; Dealloc ; free the block if no load
24512                                   ;;mov   es,[cs:area]
24513                                   ;mov    es,[area]
24514 00000776 CD21                     int     21h             ; DOS - 2+ - FREE MEMORY
24515                                                           ; ES = segment address of area to be freed
24516 00000778 EB2C                     jmp     short ProcessConfig
24517
24518                                 wfk2s_7:
24519                                   ;mov    bx,[cs:memhi]
24520                                   ;mov    es,[cs:area]
24521                                   ;sub    bx,[cs:area]            ; get desired block size in paras
24522                                   ; 07/04/2024 - Retro DOS v5.0
24523                                   ; ds = cs ; *
24524 0000077A 8CC3                     mov     bx,es
24525 0000077C F7DB                     neg     bx              ; bx = - [cs:area]
24526 0000077E 031E[6403]               add     bx,[memhi]      ; bx = [cs:memhi] - [cs:area]
24527
24528 00000782 B44A                     mov     ah,4Ah
24529 00000784 CD21                     int     21h             ; DOS - 2+ - ADJUST MEMORY BLOCK SIZE (SETBLOCK)
24530                                                           ; ES = segment address of block to change
24531                                                           ; BX = new size in paragraphs
24532 00000786 8CC0                     mov     ax,es
24533 00000788 48                       dec     ax
24534 00000789 8EC0                     mov     es,ax                   ; get Magicdrv arena
24535
24536 0000078B 26C70601000800           mov     word [es:1],8  ; [es:arena_owner]
24537                                   ;mov    word [es:ARENA.OWNER],8       ; set impossible owner
24538 00000792 26C70608005344           mov     word [es:8],4453h ; [es:arena_name],'SD' ; System Data
24539                                   ;mov    word [es:ARENA.NAME],'SD' ; 4453h
24540 00000799 2603060300                   add ax,[es:3]               ; get MCB length
24541                                   ;add    ax,[es:ARENA.SIZE]
24542
```

```
                                    ;lds    si,[cs:DOSINFO]             ; get to arena header
0000079E C536[6D02]                 lds     si,[DOSINFO]
000007A2 40                             inc     ax                  ; get addr of next MCB
000007A3 8944FE                         mov     [si-2], ax          ; store that

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

                                ; MSDOS 6.21 IO.SYS, SYSINIT:0744h

                                ; 23/03/2019 - Retro DOS v4.0 (MSDOS 6.0, SYSINIT1.ASM, 1991)
                                ; ----------------------------------------------------------------------------
                                ; 22/10/2022 - Retro DOS v4.0 (MSDOS 5.0 IO.SYS SYSINIT)
                                ; ----------------------------------------------------------------------------
                                ; 30/12/2022 - Retro DOS v4.2 (MSDOS 6.21 IO.SYS SYSINIT)
                                ; ----------------------------------------------------------------------------
                                ; 07/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM SYSINIT)

                                ; (MSDOS 6.22 IO.SYS - SYSINIT:0744h)

                                ; PCDOS 7.1 IBMBIO.COM - SYSINIT:0820h

                                ProcessConfig:
                                    ;; ds = cs ; 27/03/2019
                                    ; 11/12/2022
                                    ; ds <> cs

                                ; (MSDOS 5.0 IO.SYS - SYSINIT:0746h)

000007A6 E8BF1C                     call    doconf                  ; do pre-scan for dos=high/low

                                    ; 11/12/2022
                                    ; 27/03/2019
                                    ; ds = cs (at return from doconf)

                                ; Now, if this is not romdos, we decide what to do with the DOS code.
                                ; It will either be relocated to low memory, above the DOS data structures,
                                ; or else it will be located in HiMem, in which case a stub with the DOS
                                ; code entry points will be located in low memory. Dos_segreinit is used
                                ; to tell the DOS data where the code has been placed, and to install the
                                ; low memory stub if necessary. If the DOS is going to go into HiMem, we
                                ; must first initialize it in its present location and load the installable
                                ; device drivers. Then, if a HiMem driver has been located, we can actually
                                ; relocate the DOS code into HiMem.
                                ;
                                ; For ROMDOS, if DOS=HIGH is indicated, then we need to call dos_segreinit
                                ; to install the low memory stub (this must be done before allowing any
                                ; device drivers to hook interrupt vectors). Otherwise, we don't need to
                                ; call dos_segreinit at all, since the interrupt vector table has already
                                ; been patched.

                                    ; 22/10/2022 - Retro DOS v4.0
                                    ; (MSDOS 5.0 IO.SYS - SYSINIT:0749h)
                                    ;cmp    byte [cs:runhigh],0   ; Did user choose to run low ?
                                    ; 11/12/2022
000007A9 803E[6C02]00               cmp     byte [runhigh],0
000007AE 7404                       je      short dont_install_stub     ; yes, don't install dos low mem stub

                                ;------ user chose to load high

                                    ; 22/10/2022
                                    ;mov    es,[cs:CURRENT_DOS_LOCATION] ; MSDOS 6.21 (& MSDOS 6.0)
                                    ; 11/12/2022
                                    ; ds = cs
                                ; 13/04/2024
                                %if 0
                                    mov     es,[CURRENT_DOS_LOCATION]
                                %endif
                                    ;mov    es,[cs:FINAL_DOS_LOCATION] ; Retro DOS v4.0
                                    ; 27/03/2019
                                    ;;mov   es,[FINAL_DOS_LOCATION]

000007B0 31C0                       xor     ax,ax                   ; ax = 0 ---> install stub

                                ; 13/04/2024
                                %if 0
                                    ; 11/12/2022
                                    ; ds = cs
                                    ;call   far [cs:dos_segreinit]; call dos segreinit
                                    call    far [dos_segreinit]
                                %endif
000007B2 EB08                       jmp     short do_multi_pass

                                ;------ User chose to load dos low

                                dont_install_stub:
                                    ; 22/10/2022
000007B4 31DB                       xor     bx,bx                   ; M012
                                                                    ; don't use int 21 call to alloc mem
000007B6 E80E03                     call    MovDOSLo                ; move it !

000007B9 B80100                     mov     ax,1                    ; dont install stub

                                ; 13/04/2024
                                %if 1
                                do_multi_pass:
                                %endif
                                    ; 11/12/2022
                                    ; ds = cs
000007BC 8E06[7302]                 mov     es,[CURRENT_DOS_LOCATION]
                                    ;mov    es,[cs:CURRENT_DOS_LOCATION] ; set_dos_final_position set it up
                                    ;;mov   es,[cs:FINAL_DOS_LOCATION]   ; Retro DOS v4.0
                                    ; 27/03/2019
                                ;do_multi_pass:
                                    ;;mov   es,[FINAL_DOS_LOCATION]

                                    ; 11/12/2022
                                    ; ds =cs
                                    ;call   far [cs:dos_segreinit]; inform dos about new seg
000007C0 FF1E[7D02]                 call    far [dos_segreinit]

                                ; 13/04/2024
                                %if 0
                                do_multi_pass:
                                %endif

000007C4 E84A02                     call    AllocFreeMem            ; allocate all the free mem
                                                                    ; & update [memhi] & [area]
                                                                    ; start of free memory.
                                    ;ifdef  dblspace_hooks
                                    ;mov    bx,0                    ; magic backdoor to place int hooks
                                    ;call   cs:MagicBackdoor
                                    ;endif

                                    ; 07/04/2024 - Retro DOS v5.0
```

```
                                    ; (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 _@@@
```

```
                                    je      short ConfigDone        ; yes, no CPM hack needed
                                    call    CPMHack                 ; make ffff:d0 same as 0:c0
                                _@@@:

                                ; We are now done with CONFIG.SYS processing

                                ConfigDone:
                                    ; 12/12/2022
                                    ; 22/10/2022
                                    ;mov    byte [cs:donotshownum],1
                                                                    ; done with config.sys.
                                                                    ; do not show line number message.
                                    ;mov    es,[cs:area]
                                    ; 12/12/2022
                                    ; ds = cs
                                    ; 27/03/2019
                                    mov     byte [donotshownum],1
                                    mov     es,[area]

                                        mov     ah,49h ; DEALLOC  ; free allocated memory for command.com
                                    int     21h
                                                    ; DOS - 2+ - FREE MEMORY
                                                    ; ES = segment address of area to be freed

                                    ; 22/10/2022
                                    ;test   word [cs:install_flag],2
                                    ;test   word [cs:install_flag],has_installed ; sysinit_base installed?
                                    ;test   byte [cs:install_flag],has_installed
                                    ; 11/12/2022
                                    ; ds = cs
                                    test    byte [install_flag],2 ; has_installed
                                    ;test   byte [install_flag],has_installed
                                    jz      short skip_free_sysinitbase ; no.

                                ; set block from the old_area with impossible_owner_size.
                                ; this will free the unnecessary sysinit_base that had been put in memory to
                                ; handle install= command.
                                    ; 12/12/2022
                                       ;push      es              ; BUGBUG 3-30-92 JeffPar: no reason to save ES
                                    ;push   bx

                                    ; 22/10/2022
                                    ;mov    es,[cs:old_area]
                                    ;mov    bx,[cs:impossible_owner_size]
                                    ; 12/12/2022
                                    ; ds = cs
                                    mov     es,[old_area]
                                    mov     bx,[impossible_owner_size]

                                    mov     ah,4Ah ; SETBLOCK
                                    int     21h
                                                    ; DOS - 2+ - ADJUST MEMORY BLOCK SIZE (SETBLOCK)
                                                    ; ES = segment address of block to change
                                                    ; BX = new size in paragraphs
                                    mov     ax,es
                                    dec     ax
                                    mov     es,ax                   ; point to arena
                                    ;mov    word [es:ARENA.OWNER],8       ; set impossible owner
                                    mov     word [es:1],8
                                    ;mov    word [es:ARENA.NAME],'SD' ; 4453h ; System Data
                                    mov     word [es:8],'SD'

                                    ; 12/12/2022
                                    ;pop    bx
                                       ;pop       es              ; BUGBUG 3-30-92 JeffPar: no reason to save ES

                                skip_free_sysinitbase:
                                    ; 22/10/2022
                                    ;cmp    byte [cs:runhigh],0
                                    ; 12/12/2022
                                    ; ds = cs
                                    cmp     byte [runhigh],0
                                    je      short _@@@_ ; 04/07/2023

                                    call    InstVDiskHeader         ; Install VDISK header (allocates some mem from DOS)

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

                                _@@@_:
                                    ; 12/12/2022
                                    ; ds = cs
                                    ; 22/10/2022
                                    ; 27/03/2019
                                    ;push   cs
                                    ;pop    ds                      ; point DS to sysinitseg

                                ; set up the parameters for command

                                ;   ; 22/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS SYSINIT)
                                ;;ifdef     MULTI_CONFIG
                                ;   mov     byte [config_cmd],0   ; set special code for query_user
                                ;       call    query_user        ; to issue the AUTOEXEC prompt
                                ;   jnc     short process_autoexec; we should process autoexec normally
                                ;   ; !!!
                                ;   or      byte [bQueryOpt],4 ; MSDOS 6.21 IO.SYS - SYSINIT:081Fh
                                ;       ; !!!
                                ;   call    disable_autoexec        ; no, we should disable it
                                ;process_autoexec:
                                ;;endif     ; !!!
                                ;   call    CheckQueryOpt  ; MSDOS 6.21 IO.SYS - SYSINIT:0827h
                                ;   ; !!!

                                    ; 22/10/2022
                                    ;mov     cl,[command_line]
                                       ;mov     ch,0
                                       ;inc     cx
                                       ;mov     si,command_line
                                    ;add     si,cx
                                       ;mov     byte [si],cr    ; cr-terminate command line

                                    ; 22/10/2022 - Retro DOS 4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:0809h)

                                    ;;;;

                                    ; 30/12/2022 - Retro DOS 4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:0813h)
                                    ; ds = cs
                                    ; push  cs
                                    ; pop   ds

                                    mov     byte [config_cmd],0   ; set special code for query_user
                                    call    query_user            ; to issue the AUTOEXEC prompt
```

```
                                    ; 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]
```

```
25039 000008CD E303                         jcxz       no_env         ; use config_wrkseg only if there's env data
25040 000008CF A1[6219]                     mov ax,[config_wrkseg]
25041
25042                                   ; 22/10/2022
25043                                   ;mov    cx,[config_envlen]
25044                                       ;jcxz      no_env         ; use config_wrkseg only if there's env data
25045                                       ;mov       ax,[config_wrkseg]
25046                                 ;no_env:
25047                                   ; 22/10/2022
25048                                   ; (MSDOS 5.0 IO.SYS SYSINIT:0841h)
25049                                 no_env:
25050                                   ; 30/12/2022
25051                                   ; (MSDOS 6.21 IO.SYS SYSINIT:0861h)
25052 000008D2 29C3                     sub     bx,ax           ; bx is size of sysinit in para
25053 000008D4 83C311                   add     bx,11h          ; add the sysinit php
25054 000008D7 29DD                     sub     bp,bx           ; sub sysinit size from amount of free memory
25055 000008D9 724B                     jc      short memerrjx ; if there isn't even this much memory, give up
25056
25057                                       ;mov       ax,(OPEN<<8)   ; open the file being execed
25058 000008DB B8003D                       mov ax,3D00h
25059 000008DE F9                       stc                     ; in case of int 24
25060 000008DF CD21                     int     21h
25061 000008E1 7271                     jc      short comerr   ; ooops
25062                                                   ; DOS - 2+ - OPEN DISK FILE WITH HANDLE
25063                                                   ; DS:DX -> ASCIZ filename
25064                                                   ; AL = access mode
25065                                                   ; 0 - read
25066                                   ; 22/10/2022
25067                                   ; (MSDOS 5.0 IO.SYS SYSINIT:0852h)
25068 000008E3 89C3                         mov     bx,ax           ; handle in bx
25069
25070                                 ;   If the standard command interpreter is being used, verify it is correct
25071
25072                                   ; 30/12/2022 - Retro DOS v4.2
25073                                   ; (MSDOS 6.21 IO.SYS, SYSINIT:0874h)
25074 000008E5 803E[2A4B]00             cmp     byte [newcmd],0       ; was a new shell selected?
25075 000008EA 7518                     jne     short skip_validation ; yes
25076                                   ; 07/04/2024 - Retro DOS v5.0
25077                                   ; (PCDOS 7.1 IBMBIO.COM, SYSINIT:098Eh)
25078 000008EC BA[A608]                 mov     dx,retry-4     ; PCDOS 7.1 IBMBIO.COM - SYSINIT:0948h
25079 000008EF B90400                   mov     cx,4           ;
25080 000008F2 B43F                     mov     ah,READ        ;
25081 000008F4 CD21                     int     21h            ;
25082 000008F6 803E[A608]E9             cmp     byte [retry-4],0E9h
25083 000008FB 7557                     jne     short comerr
25084                                   ; 20/04/2019 - Retro DOS v4.0
25085                                   ; 30/12/2022
25086                                   ;cmp    byte [retry-1],64h ; MSDOS 6.21 IO.SYS - SYSINIT:088Ch
25087                                                              ; .. COMMAND.COM Version 6.20 (14h&0Fh)
25088                                   ; 07/04/2024 - Retro DOS v5.0
25089                                   ;;cmp   byte [retry-1],66h ; .. COMMAND.COM Version 6.22 (16h&0Fh)
25090                                   ;cmp    byte [retry-1],7Ah ; PCDOS 7.1 IBMBIO.COM - SYSINIT:099Fh
25091                                                              ; .. COMMAND.COM Version 7.10 (0Ah&0Fh)
25092 000008FD 803E[A908]7A             cmp     byte [retry-1],((MAJOR_VERSION&0Fh)<<4)|(MINOR_VERSION&0Fh)
25093 00000902 7550                     jne     short comerr   ;
25094
25095                                   ; 22/10/2022
25096                                   ;cmp    byte [newcmd],0       ; was a new shell selected?
25097                                   ;jne    short skip_validation ; yes
25098                                   ;mov    dx,retry-4
25099                                   ;mov    cx,4           ;
25100                                   ;mov    ah,READ        ;
25101                                   ;int    21h            ;
25102                                   ;cmp    byte [retry-4],0E9h
25103                                   ;jne    short comerr
25104                                   ;; 20/04/2019 - Retro DOS v4.0
25105                                   ;cmp    byte [retry-1],64h ; MSDOS 6.21 IO.SYS - SYSINIT:088Ch
25106                                   ;;cmp   byte [retry-1],((MAJOR_VERSION&0Fh)<<4)|(MINOR_VERSION&0Fh)
25107                                   ;jne    short comerr   ;
25108
25109                                 ;skip_validation:
25110                                   ; 22/10/2022
25111                                   ; (MSDOS 5.0 IO.SYS SYSINIT:0854h)
25112                                 skip_validation:
25113                                   ; 30/12/2022
25114                                   ; (MSDOS 6.21 IO.SYS SYSINIT:0893h)
25115 00000904 31C9                     xor     cx,cx
25116 00000906 31D2                     xor     dx,dx
25117                                   ;mov    ax,(LSEEK<<8)|2
25118 00000908 B80242                   mov     ax,4202h
25119 0000090B F9                       stc                     ; in case of int 24
25120 0000090C CD21                     int     21h             ; get file size in dx:ax
25121 0000090E 7244                     jc      short comerr
25122                                                           ; convert size in dx:ax to para in ax
25123 00000910 83C00F                   add     ax,15           ; round up size for conversion to para
25124 00000913 83D200                   adc     dx,0
25125 00000916 E88104                   call    off_to_para
25126 00000919 B10C                     mov     cl,12
25127 0000091B D3E2                     shl     dx,cl           ; low nibble of dx to high nibble
25128 0000091D 09D0                     or      ax,dx           ; ax is now # of para for file
25129 0000091F 83C010                   add     ax,10h          ; 100h byte php
25130 00000922 39E8                     cmp     ax,bp           ; will command fit in available mem?
25131 00000924 7208                     jb      short okld      ; jump if yes.
25132
25133                                 ; 30/12/2022
25134                                 %if 0
25135                                   ; 22/10/2022
25136                                 memerrjx:   ; (MSDOS 5.0 IO.SYS SYSINIT:0876h)
25137                                   ;jmp    memerr  ; (MSDOS 5.0 IO.SYS SYSINIT:34D5h)
25138                                   ; 02/11/2022
25139                                   ;jmp    mem_err
25140                                   ; 11/12/2022
25141                                   ; ds = cs
25142                                   jmp     mem_err2
25143                                 %endif
25144                                   ; 30/12/2022
25145                                   ; (MSDOS 6.21, IO.SYS, SYSINIT:08B5h)
25146                                 memerrjx:
25147 00000926 BA[4951]                 mov     dx,badmem      ; "Configuration too large for memory"
25148 00000929 E82841                   call    print
25149 0000092C EB3A                     jmp     short continue
25150
25151                                 okld:
25152 0000092E B43E                     mov     ah,3Eh ; CLOSE
25153 00000930 CD21                     int     21h            ; close file
25154
25155                                   ; 22/10/2022
25156 00000932 5A                       pop     dx      ; (MSDOS 5.0 IO.SYS SYSINIT:087Dh)
25157
25158                                   ; 24/03/2019
25159
25160 00000933 0E                        push    cs             ; point es to sysinitseg
25161 00000934 07                       pop     es
25162 00000935 BB[BF02]                     mov     bx,COMEXE ; point to exec block
```

```
                                    ; 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:
00000985 BE[7E4B]                       mov     si,commnd3
00000988 803C00                         cmp     byte [si],0     ; has 3rd alternate been tried?
0000098B 754C                           jne     short do_alt    ; no
0000098D BE[9E4B]                       mov     si,commnd4
00000990 803C00                         cmp     byte [si],0     ; has 4th alternate been tried?
00000993 7544                           jne     short do_alt    ; no
00000995 52                             push    dx              ;
00000996 BA[3853]                       mov     dx,badcomprmpt
00000999 E8B840                         call    print           ;
0000099C 5A                             pop     dx              ; recover default drive letter in DL
                                    request_input:                  ;
0000099D B402                           mov     ah,STD_CON_OUTPUT
0000099F CD21                           int     21h             ;
000009A1 52                             push    dx              ;
000009A2 B23E                           mov     dl,'>'          ;
000009A4 CD21                           int     21h             ;
000009A6 8A1E[2C4B]                     mov     bl,[tmplate+1] ; [tmplate+1] = 12
000009AA B700                           mov     bh,0            ;
000009AC C687[2D4B]0D                   mov     byte [commnd+bx],0Dh
000009B1 BA[2B4B]                       mov     dx,tmplate
000009B4 B40A                           mov     ah,STD_CON_STRING_INPUT
000009B6 CD21                           int     21h             ; read a line of input
000009B8 BA[7050]                       mov     dx,crlfm        ;
000009BB E89640                         call    print           ;
000009BE 5A                             pop     dx              ;
000009BF 8A1E[2C4B]                     mov     bl,[tmplate+1] ;
000009C3 08DB                           or      bl,bl           ; was anything typed?
000009C5 74D6                           jz      short request_input ;
000009C7 C606[2A4B]01                   mov     byte [newcmd],1 ; disable validation for user-specified binaries
000009CC C687[2D4B]00                   mov     byte [commnd+bx],0 ; NULL-terminate it before execing it
000009D1 C706[BB4B]000D                 mov     word [command_line],0D00h
000009D7 EB35                           jmp     short do_exec   ;
                                    do_alt:
000009D9 1E                             push    ds
000009DA 07                             pop     es
000009DB C606[2A4B]00                   mov     byte [newcmd],0 ; force validation for alternate binaries
000009E0 BF[2D4B]                       mov     di,commnd       ;
                                    do_alt1:
000009E3 AC                             lodsb                   ; copy the alternate, zapping it as we go,
000009E4 C644FF00                       mov     byte [si-1],0   ; so that we know it's been tried
000009E8 AA                             stosb                   ;
000009E9 08C0                           or      al,al           ;
000009EB 75F6                           jnz     short do_alt1   ;
000009ED BF[BB4B]                       mov     di,command_line
000009F0 807C023A                       cmp     byte [si+2],':'
000009F4 7503                           jne     short do_alt2   ;
000009F6 885401                         mov     [si+1],dl       ; stuff default drive into alt. command line
                                    do_alt2:                    ;
000009F9 AC                             lodsb                   ;
000009FA AA                             stosb                   ;
000009FB 08C0                           or      al,al           ;
000009FD 75FA                           jnz     short do_alt2   ;
000009FF C645FF0D                       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-
```

```
25405 00000A5A 7420616C6C6F636174-
25405 00000A63 65204D656D6F727920-
25405 00000A6C 666F7220444F530D0A-
25405 00000A75 24
25406
25407                                  ; --------------------------------------------------------------------
25408                                  ;
25409                                  ; procedure : LoadDOSHiOrLo
25410                                  ;
25411                                  ;         Tries to move DOS into HMA. If it fails then loads
25412                                  ;         DOS into Low memory. For ROMDOS, nothing is actually
25413                                  ;         moved; this just tries to allocate the HMA, and prints
25414                                  ;         a message if this is not possible.
25415                                  ;
25416                                  ; --------------------------------------------------------------------
25417
25418                                      ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
25419                                  LoadDOSHiOrLo:
25420                                      ; 27/03/2019 - Retro DOS v4.0
25421                                      ; ds = cs
25422 00000A76 E81F00                    call    TryToMovDOSHi         ; Try moving it into HMA (M024)
25423                                    ;jc     short LdngLo          ; If that don't work...
25424                                    ;retn
25425                                      ; 18/12/2022
25426 00000A79 731C                    jnc     short LoadDosHi_ok
25427                                  LdngLo:
25428                                      ; 23/10/2022
25429                                    ;push   cs
25430                                    ;pop    ds
25431                                      ; 11/12/2022
25432                                      ; ds = cs
25433 00000A7B B409                    mov     ah,9
25434 00000A7D BA[230A]                mov     dx,DOSLOMSG           ; inform user that we are
25435 00000A80 CD21                    int     21h                   ; loading low
25436
25437                                  ;ifndef ROMDOS
25438                                      ; actually move the dos, and reinitialize it.
25439
25440 00000A82 BB0100                  mov     bx,1                          ; M012
25441                                                                        ;  use int 21 alloc for mem
25442 00000A85 E83F00                  call    MovDOSLo
25443                                    ; 11/12/2022
25444                                    ; ds = cs
25445                                    ;mov    es,[cs:CURRENT_DOS_LOCATION]  ; give dos its temporary loc.
25446                                    ; 23/10/2022
25447 00000A88 8E06[7302]              mov     es,[CURRENT_DOS_LOCATION]
25448                                    ;;mov   es,[cs:FINAL_DOS_LOCATION]  ; 24/03/2019 - Retro DOS v4.0
25449                                    ;mov    es,[FINAL_DOS_LOCATION] ; 27/03/2019
25450 00000A8C 31C0                    xor     ax,ax                         ; ax = 00 ---> install stub
25451                                    ; 11/12/2022
25452                                    ; ds = cs
25453                                    ;call   far [cs:dos_segreinit]        ; call dos segreinit
25454 00000A8E FF1E[7D02]              call    far [dos_segreinit] ; 27/03/2019
25455
25456                                  ;endif ; ROMDOS
25457                                      ; 23/10/2022
25458                                    ;mov    byte [cs:runhigh],0           ; mark that we are running lo
25459                                      ; 11/12/2022
25460                                      ; ds = cs
25461 00000A92 C606[6C02]00            mov     byte [runhigh],0 ; 27/03/2019
25462                                  LoadDosHi_ok:       ; 18/12/2022
25463 00000A97 C3                      retn
25464
25465                                  ; --------------------------------------------------------------------
25466                                  ;
25467                                  ; procedure : TryToMovDOSHi
25468                                  ;
25469                                  ;         This tries to move DOS into HMA.
25470                                  ;         Returns CY if it failed.
25471                                  ;         If it succeeds returns with carry cleared.
25472                                  ;
25473                                  ;         For ROMDOS, dos_segreinit must be called again to allow
25474                                  ;         the A20 switching code in the low mem stub to be installed.
25475                                  ;
25476                                  ; --------------------------------------------------------------------
25477
25478                                      ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
25479                                      ; (MSDOS 5.0 IO.SYS - SYSINIT:092Ah)
25480                                  TryToMovDOSHi:
25481                                      ; 11/12/2022
25482                                      ; 27/03/2019 - Retro DOS v4.0
25483                                      ; ds = cs
25484 00000A98 E81300                  call    MovDOSHi
25485 00000A9B 7210                    jc      short ttldhx
25486
25487                                  ;ifndef ROMDOS
25488                                      ; 23/10/2022
25489                                    ;mov    es,[cs:CURRENT_DOS_LOCATION] ; give dos its temporary loc.
25490                                    ;;mov   es,[cs:FINAL_DOS_LOCATION] ; 24/03/2019 - Retro DOS v4.0
25491                                      ; 11/12/2022
25492                                      ; ds = cs
25493 00000A9D 8E06[7302]              mov     es,[CURRENT_DOS_LOCATION]
25494                                  ;else
25495                                  ;       ..
25496                                  ;endif ; ROMDOS
25497
25498                                      ; 11/12/2022
25499                                      ; ds = cs
25500 00000AA1 31C0                    xor     ax,ax                 ; ax = 00 ---> install stub
25501                                    ;call   far [cs:dos_segreinit]; call dos segreinit
25502 00000AA3 FF1E[7D02]              call    far [dos_segreinit]
25503                                    ;mov    byte [cs:runhigh],1
25504 00000AA7 C606[6C02]01            mov     byte [runhigh],1
25505 00000AAC F8                      clc
25506                                  ttldhx:
25507 00000AAD C3                      retn
25508
25509                                  ; --------------------------------------------------------------------
25510                                  ;
25511                                  ; procedure : MovDOSHi
25512                                  ;
25513                                  ;         Tries to allocate HMA and Move DOS/BIOS code into HMA
25514                                  ;         For ROMDOS, the code is not actually moved, but the
25515                                  ;         HMA is allocated and prepared for sub-allocation.
25516                                  ;
25517                                  ;         Returns : CY if it failed
25518                                  ;
25519                                  ; --------------------------------------------------------------------
25520
25521                                      ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
25522                                  MovDOSHi:
25523                                      ; 14/05/2019
25524                                      ; 27/03/2019 - Retro DOS v4.0
25525                                      ; ds = cs
```

```
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
```

```
26022 00000B92 B81043                   mov     ax,4310h
26023 00000B95 CD2F                     int     2Fh             ; get the vector into es:bx
26024                                           ; - Multiplex - XMS - GET DRIVER ADDRESS
26025                                           ; Return: ES:BX -> driver entry point
26026
26027 00000B97 891E[0E00]               mov     [xms],bx
26028                                   ;mov    [0Eh], bx
26029 00000B9B 8C06[1000]               mov     [xms+2],es
26030                                   ;mov    [10h],es
26031
26032 00000B9F B401                     mov     ah,1            ; request HMA
26033 00000BA1 BAFFFF                   mov     dx,0FFFFh
26034                                   ;call   dword ptr ds:0Eh
26035 00000BA4 FF1E[0E00]               call    far [xms]
26036 00000BA8 48                       dec     ax
26037 00000BA9 7409                     jz      short allocHMA_1 ; error if not able to allocate HMA
26038
26039                                 ;------ Himem may be lying because it has allocated mem for int 15
26040
26041 00000BAB B488                     mov     ah,88h
26042 00000BAD CD15                     int     15h
26043                                           ; Get Extended Memory Size
26044                                           ; Return: CF clear on success
26045                                           ; AX = size of memory above 1M in K
26046 00000BAF 83F840                   cmp     ax,64           ; less than 64 K of hma ?
26047                                   ;jb     short grabhma_error
26048                                   ; 11/12/2022
26049 00000BB2 7224                     jb      short grabhma_err ; cf=1
26050                                 allocHMA_1:
26051 00000BB4 B405                     mov     ah,5            ; localenableA20
26052                                   ;call   dword ptr ds:0Eh
26053 00000BB6 FF1E[0E00]               call    far [xms]
26054 00000BBA 48                       dec     ax
26055 00000BBB 751A                     jnz     short grabhma_error ; error if couldn't enable A20
26056
26057 00000BBD E89D01                   call    IsVDiskInstalled
26058 00000BC0 7415                     jz      short grabhma_error ; yes, we cant use HMA
26059
26060 00000BC2 B8FFFF                   mov     ax,0FFFFh
26061 00000BC5 8EC0                     mov     es,ax
26062 00000BC7 26C70610003412           mov     word [es:10h],1234h ; see if we can really read/write there
26063 00000BCE 26813E10003412           cmp     word [es:10h],1234h
26064                                   ;jne    short grabhma_error ; don't try to load there if XMS lied
26065                                   ; 11/12/2022
26066 00000BD5 7401                     je      short allocHMA_ok
26067
26068                                 ; 11/12/2022
26069                                 ;   ; 11/12/2022
26070                                 ;   ; cf=0
26071                                 ;   ;clc
26072                                 ;   pop     ds
26073                                 ;   retn
26074
26075                                 grabhma_error:
26076 00000BD7 F9                       stc
26077                                   ; 11/12/022
26078                                 grabhma_err:        ; cf=1
26079                                 allocHMA_ok:        ; cf=0
26080 00000BD8 1F                       pop     ds
26081 00000BD9 C3                       retn
26082
26083                                 ; ---------------------------------------------------------------------------
26084                                 ;
26085                                 ; procedure : IsXMSLoaded
26086                                 ;
26087                                 ;             Checks whether a XMS driver is loaded
26088                                 ;
26089                                 ; Returns : Z flag set if XMS driver loaded
26090                                 ;           Z flag reset if no XMS drivers are present
26091                                 ;
26092                                 ; ---------------------------------------------------------------------------
26093
26094                                 IsXMSLoaded:
26095 00000BDA B80043                   mov     ax,4300h
26096 00000BDD CD2F                     int     2Fh             ; - Multiplex - XMS - INSTALLATION CHECK
26097                                                           ; Return: AL = 80h XMS driver installed
26098                                                           ; AL <> 80h no driver
26099 00000BDF 3C80                     cmp     al,80h          ; XMS installed?
26100 00000BE1 C3                       retn
26101
26102                                 ; ---------------------------------------------------------------------------
26103                                 ; procedure : FTryToMovDOSHi
26104                                 ;
26105                                 ;           Called from HMA suballoc calls
26106                                 ;
26107                                 ; ---------------------------------------------------------------------------
26108
26109                                     ; 23/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
26110                                     ; (MSDOS 5.0 IO.SYS - SYSINIT:0A84h)
26111
26112                                     ; 08/04/2024 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
26113                                     ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:0C9Fh)
26114
26115                                     ; ((MSDOS 6.22 IO.SYS - SYSINIT:0B8Ch))
26116
26117                                 FTryToMovDOSHi:    ; proc  far
26118
26119 00000BE2 50                       push    ax
26120 00000BE3 53                       push    bx
26121 00000BE4 51                       push    cx
26122 00000BE5 52                       push    dx
26123 00000BE6 56                       push    si
26124 00000BE7 57                       push    di
26125 00000BE8 1E                       push    ds
26126 00000BE9 06                       push    es
26127
26128                                   ; 23/10/2022
26129                                   ; 27/03/2019 - Retro DOS v4.0
26130                                   ; 11/12/2022
26131 00000BEA 0E                       push    cs
26132 00000BEB 1F                       pop     ds
26133
26134                                   ;cmp    byte [cs:runhigh],0FFh
26135                                   ; 11/12/2022
26136 00000BEC 803E[6C02]FF             cmp     byte [runhigh],0FFh
26137 00000BF1 7503                     jne     short _ftymdh_1
26138
26139                                   ; ds = cs
26140 00000BF3 E8A2FE                   call    TryToMovDOSHi
26141                                 _ftymdh_1:
26142 00000BF6 07                       pop     es
26143 00000BF7 1F                       pop     ds
26144 00000BF8 5F                       pop     di
26145 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
```

```
26642 00000DC2 89F0                     mov     ax,si
26643 00000DC4 26C745160000             mov     word [es:di+SYSI_CDS],0       ; set address of CDS list
26644                                   ;mov    [word es:di+16h],0
26645                                   ;lds    si,[es:di+SYSI_DPB]   ; (ds:si) = address of first DPB
26646 00000DCA 26C535                   lds     si,[es:di]
26647 00000DCD 8EC0                     mov     es,ax
26648 00000DCF 31FF                     xor     di,di                 ; (es:di) = address of 1st CDS
26649
26650                                 ;*  Initialize our temporary CDSs. We'll init each CDS with the
26651                                 ;   info from the corresponding DPB.
26652                                 ;
26653                                 ;   (cx) = count of CDSs left to process
26654                                 ;   (es:di) = address of next CDS
26655
26656                                 fooset:
26657                                     ; 23/10/2022
26658 00000DD1 2EA1[A902]               mov     ax,[cs:DirStrng] ; "A:"
26659 00000DD5 AB                       stosw                         ; setup the root as the curdir
26660
26661                                     ; 23/10/2022 (MSDOS 5.0 SYSINIT)
26662                                     ;call   get_dpb_for_drive_al  ; get dpb for drive in dpb
26663
26664                                     ; 30/12/2022
26665                                     ; (MSDOS 6.21 SYSINIT:0D8Bh)
26666 00000DD6 E85200                   call    get_dpb_for_drive_al  ; get dpb for drive in dpb
26667
26668                                 ;   (ds:si) = address of DPB
26669                                 ;            (si) = -1 if no drive
26670
26671 00000DD9 2EA1[AB02]               mov     ax,[cs:DirStrng+2] ; "\",0
26672 00000DDD AB                       stosw
26673 00000DDE 2EFE06[A902]             inc     byte [cs:DirStrng]
26674 00000DE3 31C0                     xor     ax,ax ; 0
26675 00000DE5 51                       push    cx
26676                                   ;mov    cx,curdir_list.cdir_flags - 4 ; 63
26677 00000DE6 B93F00                   mov     cx,63   ; 23/10/2022
26678 00000DE9 F3AA                     rep     stosb                 ; zero out rest of CURDIR_TEXTs
26679
26680                                 ;   should handle the system that does not have any floppies.
26681                                 ;   in this case,we are going to pretended there are two dummy floppies
26682                                 ;   in the system. still they have dpb and cds,but we are going to
26683                                 ;   0 out curdir_flags,curdir_devptr of cds so ibmdos can issue
26684                                 ;   "invalid drive specification" message when the user try to
26685                                 ;   access them.
26686                                 ;
26687                                 ;   (ax) = 0
26688                                 ;   (es:di) = CURDIR_FLAGS in the CDS records
26689                                 ;   (ds:si) = Next DPB (-1 if none)
26690
26691 00000DEB 83FEFF                     cmp     si,-1   ; cmp si,0FFFFh
26692 00000DEE 740C                     je      short fooset_zero     ; don't have any physical drive.
26693
26694                                 ;   check to see if we are faking floppy drives. if not go to normcds.
26695                                 ;   if we are faking floppy drives then see if this cds being initialised
26696                                 ;   is for drive a: or b: by checking the appropriate field in the dpb
26697                                 ;   pointed to by ds:si. if not for a: or b: then go to normcds. if
26698                                 ;   for a: or b: then execute the code given below starting at fooset_zero.
26699                                 ;   for dpb offsets look at inc\dpb.inc.
26700
26701                                     ; 03/09/2023
26702 00000DF0 41                       inc     cx  ; cx = 1
26703
26704 00000DF1 2E380E[8B02]             cmp     [cs:fake_floppy_drv],cl ; 1 ; 03/09/2023
26705                                   ;cmp    byte [cs:fake_floppy_drv],1
26706 00000DF6 750A                     jne     short normcds         ; machine has floppy drives
26707                                   ;cmp    byte [si+DPB.drive],1 ; if dpb_drive = 0 (a) or 1 (b).
26708                                   ;cmp    byte [si],1
26709 00000DF8 380C                     cmp     [si],cl ; 1 ; 03/09/2023
26710 00000DFA 7706                     ja      short normcds
26711
26712                                     ; 30/12/2023
26713                                     ; ax = 0
26714                                 fooset_zero:
26715 00000DFC B103                     mov     cl,3                  ; the next dbp pointer
26716                                                                 ; AX should be zero here
26717 00000DFE F3AB                     rep     stosw
26718                                   ; 30/12/2023
26719                                   ;pop    cx
26720 00000E00 EB0F                     jmp     short get_next_dpb ; findcds
26721
26722                                 ;   (ax) = 0
26723
26724                                     ; 30/12/2023
26725                                 ;fooset_zero:
26726                                   ;mov    cl,3
26727                                   ;rep    stosw
26728                                   ;pop    cx
26729                                   ;jmp    short fincds
26730
26731                                 ;*  We have a "normal" DPB and thus a normal CDS.
26732                                 ;
26733                                 ;   (ax) = 0
26734                                 ;   (es:di) = CURDIR_FLAGS in the CDS records
26735                                 ;   (ds:si) = Next DPB (-1 if none)
26736
26737                                 normcds:
26738                                     ; 30/12/2023
26739                                     ;pop    cx
26740
26741                                 ;   if a non-fat based media is detected (by dpb.numberoffat == 0), then
26742                                 ;   set curdir_flags to 0. this is for signaling ibmdos and ifsfunc that
26743                                 ;   this media is a non-fat based one.
26744
26745                                     ;cmp    byte [si+DPB.FAT_COUNT],0 ; non fat system?
26746                                     ; 23/10/2022
26747                                     ;cmp    byte [si+8],0
26748                                     ; 03/09/2023 (ax=0)
26749 00000E02 384408                   cmp     [si+8],al ; 0
26750 00000E05 7403                     je      short setnormcds      ; yes. set curdir_flags to 0. ax = 0 now.
26751 00000E07 B80040                   mov     ax,curdir_inuse ; 4000h       ; else,fat system. set the flag to curdir_inuse.
26752                                   ;mov    ax,4000h
26753                                 setnormcds:
26754 00000E0A AB                       stosw                         ; curdir_flags
26755 00000E0B 89F0                     mov     ax,si
26756 00000E0D AB                       stosw                         ; curdir_devptr
26757 00000E0E 8CD8                     mov     ax,ds
26758 00000E10 AB                       stosw
26759
26760                                 get_next_dpb:                           ; entry point for fake_fooset_zero
26761                                     ; 30/12/2022
26762                                     ; (MSDOS 6.21 SYSINIT:0DD1h)
26763                                     ; 23/10/2022
26764                                     ;lds    si,[si+19h] ; (MSDOS 5.0 SYSINIT)
26765                                     ;;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:
```

```
27386 00001073 A1[8C02]                 mov     ax,[stack_count]        ; stack_count = 0?
27387 00001076 09C0                     or      ax,ax                   ; then, stack size must be 0 too.
27388 00001078 7466                     jz      short skipstack                 ; don't install stack.
27389
27390                                ;   dynamic relocation of stack code.
27391
27392 0000107A E87537                   call    round                   ;[memhi] = seg. for stack code
27393                                                                   ;[memlo] = 0
27394
27395                                ; set devmark block into memory for mem command
27396                                ; devmark_id = 's' for stack
27397
27398                                    ;mov    al,devmark_stk ;='S'
27399                                    ; 23/10/2022
27400 0000107D B053                     mov     al,'S'
27401 0000107F E80906                   call    setdevmark
27402
27403 00001082 A1[6403]                 mov     ax,[memhi]
27404 00001085 8EC0                     mov     es,ax           ;es -> seg. the stack code is going to move.
27405                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
27406                                   ; 11/12/2022
27407                                   ; ds = cs
27408                                   ;push   cs
27409                                   ;pop    ds
27410 00001087 31F6                     xor     si,si           ;!!we know that stack code is at the beginning of sysinit.
27411 00001089 31FF                     xor     di,di
27412 0000108B B9[6902]                 mov     cx,endstackcode
27413 0000108E 890E[6203]               mov     [memlo],cx
27414 00001092 E85D37                   call    round           ;have enough space for relocation?
27415 00001095 F3A4                     rep     movsb
27416
27417 00001097 1E                       push    ds              ; stick the location of the NextStack entry
27418                                   ;;mov   ax,Bios_Data    ; into the Win386 Instance Data tables
27419                                   ;mov    ax,KERNEL_SEGMENT ; 70h
27420                                   ; 21/10/2022
27421 00001098 B87000                   mov     ax,DOSBIODATASEG ; 0070h
27422 0000109B 8ED8                     mov     ds,ax
27423 0000109D C706[0208][1000]         mov     word [NextStack],nextentry ; (8C0h for MSDOS 6.21 IO.SYS)
27424 000010A3 8C06[0408]               mov     [NextStack+2],es          ; (8C2h for MSDOS 6.21 IO.SYS)
27425
27426 000010A7 2EA1[6203]               mov     ax,[cs:memlo]
27427 000010AB 2EA3[9002]               mov     [cs:stack_addr],ax ;set for stack area initialization
27428 000010AF A3[0808]                 mov     [IT_StackLoc],ax ; pass it as Instance Data, too
27429 000010B2 2EA1[6403]               mov     ax,[cs:memhi]  ;this will be used by stack_init routine.
27430 000010B6 2EA3[9202]               mov     [cs:stack_addr+2],ax
27431 000010BA A3[0A08]                 mov     [IT_StackLoc+2],ax
27432
27433                                ;   space for internal stack area = stack_count(entrysize + stack_size)
27434
27435                                    ;mov    ax,entrysize ; mov ax,8
27436                                    ; 23/10/2022
27437 000010BD B80800                   mov     ax,8
27438 000010C0 2E0306[8E02]             add     ax,[cs:stack_size]
27439 000010C5 2EF726[8C02]             mul     word [cs:stack_count]
27440
27441 000010CA A3[0C08]                 mov     [IT_StackSize],ax ; pass through to Instance Tables
27442
27443 000010CD 1F                       pop     ds              ; no more need to access Instance Table
27444
27445 000010CE E84402                   call    ParaRound       ; convert size to paragraphs
27446
27447                                   ; 11/12/2022
27448                                   ; ds = cs
27449                                   ;add    [cs:memhi],ax
27450 000010D1 0106[6403]               add     [memhi],ax
27451                                   ;or     byte [cs:setdevmarkflag],for_devmark ; 2
27452                                   ;or     byte [cs:setdevmarkflag],2
27453 000010D5 800E[6919]02             or      byte [setdevmarkflag],2
27454                                   ;or     byte [setdevmarkflag],for_devmark ; 2
27455                                                           ;to set the devmark_size for stack by round routine.
27456 000010DA E81537                   call    round           ; check for memory error before
27457                                                           ; continuing
27458 000010DD E87D03                   call    stackinit       ; initialize hardware stack.
27459                                                           ; cs=ds=sysinitseg,es=relocated stack code & data
27460                                skipstack:
27461
27462                                   ; 10/04/2024 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
27463                                   ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:11F0h)
27464                                   ;;;
27465                                   ;push   cs
27466                                   ;pop    ds
27467                                   ; ds = cs
27468 000010E0 803E[6E03]01             cmp     byte [dosdata_umb],1  ; PCDOS 7 feature - DOSDATA=UMB/NOUMB configuration
27469                                                                 ; 1 = DOSDATA=UMB, 2 = (UMB) done, 0 = NOUMB
27470 000010E5 7773                     ja      short dosdata_umb_done; 2 - done
27471 000010E7 727D                     jb      short dosdata_noumb   ; 0 - DOSDATA=NOUMB
27472
27473 000010E9 803E[8B16]EA             cmp     byte [setdevmark],0EAh
27474 000010EE 7476                     je      short dosdata_noumb
27475
27476 000010F0 B80258                   mov     ax,5802h
27477 000010F3 CD21                     int     21h             ; DOS - 3+ - GET/SET MEMORY ALLOCATION STRATEGY
27478                                                           ; AL = function code: (DOS 5beta) get UMB link state
27479 000010F5 98                       cbw
27480 000010F6 89C7                     mov     di,ax                   ; al = 01h -> UMBs in DOS memory chain
27481                                                                   ; save current (previous) UMB link state
27482 000010F8 BB0100                   mov     bx,1                    ; bx = 01h -> add UMBs to DOS memory chain
27483
27484 000010FB B80358                   mov     ax,5803h
27485 000010FE CD21                     int     21h
27486 00001100 7264                     jc      short dosdata_noumb
27487
27488 00001102 B80058                   mov     ax,5800h
27489 00001105 CD21                     int     21h             ; DOS - 3+ - GET/SET MEMORY ALLOCATION STRATEGY
27490                                                           ; AL = function code: get allocation strategy
27491
27492 00001107 89C6                     mov     si,ax                   ; ax = current strategy
27493                                                                   ; save current (previous) allocation strategy
27494 00001109 BB4000                   mov     bx,40h                  ; bl = new strategy = 40h - high memory first fit
27495
27496 0000110C B80158                   mov     ax,5801h
27497 0000110F CD21                     int     21h
27498
27499 00001111 8B1E[6403]               mov     bx,[memhi]
27500 00001115 2B1E[6A03]               sub     bx,[prev_memhi]
27501
27502 00001119 B448                      mov     ah,48h
27503 0000111B CD21                     int     21h             ; DOS - 2+ - ALLOCATE MEMORY
27504                                                           ; BX = number of 16-byte paragraphs desired
27505 0000111D 89C1                     mov     cx,ax                   ; ax = segment of allocated block
27506 0000111F 89FB                     mov     bx,di                   ; restore previous UMB link state
27507
27508 00001121 B80358                   mov     ax,5803h
27509 00001124 CD21                     int     21h             ; DOS - 3+ - GET/SET MEMORY ALLOCATION STRATEGY
```

```
                                                        ; 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
```

```
27634
27635 000011B5 B0FF                      mov     al,0FFh             ;reset h/w by writing to port
27636 000011B7 BAF202                    mov     dx,2F2h             ;get starting address
27637 000011BA EE                        out     dx,al               ; out 02f2h,0ffh
27638 000011BB 42                        inc     dx
27639 000011BC EE                        out     dx,al               ; out 02f3h,0ffh
27640 000011BD 42                        inc     dx
27641 000011BE EE                        out     dx,al               ; out 02f4h,0ffh
27642 000011BF 42                        inc     dx
27643 000011C0 EE                        out     dx,al               ; out 02f5h,0ffh
27644 000011C1 42                        inc     dx
27645 000011C2 EE                        out     dx,al               ; out 02f6h,0ffh
27646 000011C3 42                        inc     dx
27647 000011C4 EE                        out     dx,al               ; out 02f7h,0ffh
27648
27649                                 ;sb secondary global rearm
27650
27651 000011C5 B800F0                    mov     ax,0F000h           ;get machine type
27652 000011C8 8EC0                      mov     es,ax
27653 000011CA 26803EFEFFFC              cmp     byte [es:0FFFEh],0FCh ;q:is it a at type machine
27654 000011D0 740D                      je      short startrearm    ; *if at no need to check
27655
27656 000011D2 B4C0                      mov     ah,0C0h             ;get system configuration
27657 000011D4 CD15                      int     15h                 ; *
27658 000011D6 7216                      jc      short finishrearm   ; *jmp if old rom
27659
27660                                 ; test feature byte for secondary interrupt controller
27661
27662 000011D8 26F6470540                test    byte [es:bx+5],40h
27663                                    ; 24/10/2022
27664                                    ;test   byte [es:bx+ROMBIOS_DESC.bios_sd_featurebyte1],ScndIntController
27665 000011DD 740F                      je      short finishrearm   ;jmp if it is there
27666
27667                                 startrearm:
27668 000011DF B0FF                      mov     al,0FFh             ;write any pattern to port
27669 000011E1 BAF206                    mov     dx,6F2h             ;get starting address
27670 000011E4 EE                        out     dx,al               ;out 06f2h,0ffh
27671 000011E5 42                        inc     dx                  ;bump address
27672 000011E6 EE                        out     dx,al               ;out 06f3h,0ffh
27673 000011E7 42                        inc     dx                  ;bump address
27674 000011E8 EE                        out     dx,al               ;out 06f4h,0ffh
27675 000011E9 42                        inc     dx                  ;bump address
27676 000011EA 42                        inc     dx                  ;bump address
27677 000011EB EE                        out     dx,al               ;out 06f6h,0ffh
27678 000011EC 42                        inc     dx                  ;bump address
27679 000011ED EE                        out     dx,al               ;out 06f7h,0ffh
27680
27681                                 finishrearm:
27682 000011EE 07                        pop     es
27683 000011EF 5A                        pop     dx
27684 000011F0 5B                        pop     bx
27685 000011F1 58                        pop     ax
27686
27687                                 ;    global rearm end ********************
27688
27689                                 ; ------------------------------------------------------------------------
27690                                 ; allocate sysinit_base for install= command
27691                                 ; ------------------------------------------------------------------------
27692                                 ; sysinit_base allocation.
27693                                 ;   check if endfile has been called to handle install= command.
27694
27695                                 set_sysinit_base:
27696
27697                                 ; ------------------------------------------------------------------------
27698                                 ;sysinit_base will be established in the secure area of
27699                                 ;lower memory when it handles the first install= command.
27700                                 ;sysinit_base is the place where the actual exec function will be called and
27701                                 ;will check sysinit module in high memory if it is damaged by the application
27702                                 ;program.  if sysinit module has been broken,then "memory error..." message
27703                                 ;is displayed by sysinit_base.
27704                                 ; ------------------------------------------------------------------------
27705
27706                                     ; 24/10/2022 - Retro DOS v4.0 (MSDOS 5.0 IO.SYS, SYSINIT)
27707                                     ; (SYSINIT:1028h)
27708
27709                                     ; 11/12/2022
27710                                     ; ds = cs
27711 000011F2 50                        push    ax                  ; set devmark for mem command
27712 000011F3 A1[6403]                  mov     ax,[memhi]
27713 000011F6 2B06[6803]                sub     ax,[area]
27714 000011FA A3[6003]                  mov     [impossible_owner_size],ax ;remember the size in case.
27715                                    ;mov    al,devmark_inst ; 'T'
27716 000011FD B054                      mov     al,'T'
27717 000011FF E88904                    call    setdevmark
27718 00001202 58                        pop     ax
27719
27720 00001203 8B3E[6403]                mov     di,[memhi]
27721 00001207 8EC7                      mov     es,di
27722 00001209 893E[D402]                mov     [sysinit_base_ptr+2],di ; save this entry for the next use.
27723 0000120D 31FF                      xor     di,di
27724 0000120F 893E[D202]                mov     [sysinit_base_ptr],di ; es:di -> destination.
27725 00001213 BE[2113]                  mov     si,sysinit_base             ;ds:si -> source code to be relocated.
27726 00001216 B98100                    mov     cx,end_sysinit_base-sysinit_base ; 129
27727                                    ; 24/10/2022
27728                                    ;mov    cx,128  ; 11DCh-115Ch  ; (MSDOS 5.0 IO.SYS, SYSINIT)
27729 00001219 010E[6203]                add     [memlo],cx
27730                                    ;or     byte cs:[setdevmarkflag],for_devmark ; 2
27731                                    ; 11/12/2022
27732                                    ; ds = cs
27733                                    ;or     byte [cs:setdevmarkflag],2
27734 0000121D 800E[6919]02              or      byte [setdevmarkflag],2
27735                                    ;or     byte [setdevmarkflag],for_devmark
27736 00001222 E8CD35                    call    round               ; check mem error. also,readjust memhi for the next use.
27737 00001225 F3A4                      rep     movsb               ; reallocate it.
27738
27739 00001227 C706[D602][0813]           mov     word [sysinit_ptr],sysinitptr ; returning address from
27740 0000122D 8C0E[D802]                mov     [sysinit_ptr+2],cs     ; sysinit_base back to sysinit.
27741                                    ;or     word [install_flag],has_installed ; set the flag.
27742                                    ;or     byte [install_flag],has_installed ; 2
27743                                    ; 11/12/2022
27744 00001231 800E[CE02]02              or      byte [install_flag],2
27745                                    ; 24/10/2022
27746                                    ;or     word [install_flag],2
27747
27748                                 ; ------------------------------------------------------------------------
27749                                 ; free the rest of the memory from memhi to confbot. still from confbot to
27750                                 ; the top of the memory will be allocated for sysinit and config.sys if
27751                                 ; have_install_cmd.
27752                                 ; ------------------------------------------------------------------------
27753
27754 00001236 E8B935                    call    round
27755 00001239 8B1E[6403]                mov     bx,[memhi]
27756 0000123D A1[6803]                  mov     ax,[area]
27757 00001240 A3[5E03]                  mov     [old_area],ax       ; save [area]
```

```
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
000012B2 268805                     mov     [es:di],al
000012B5 3C0A                       cmp     al,lf  ; cmp al,0Ah   ; line feed?
000012B7 7405                       je      short done_installparm
000012B9 FEC1                       inc     cl                    ; # of char. in the parm.
000012BB 47                         inc     di
000012BC EBF3                       jmp     short got_installparm
                                done_installparm:
000012BE 2E880E[F002]               mov     byte [cs:ldexec_line],cl ; length of the parm.
                                    ; 05/09/2023
000012C3 08C9                       or      cl,cl
                                    ;cmp    cl,0                  ; if no parm,then
000012C5 7506                       jne     short install_seg_set ; let the parm area
000012C7 2EC606[F102]0D             mov     byte [cs:ldexec_start],cr ; 0Dh
                                                                  ; starts with cr.
                                install_seg_set:
                                    ; 05/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
000012CD 31DB                       xor     bx,bx
                                    ;mov    word [cs:0],0         ; make a null environment segment
000012CF 2E891F                     mov     [cs:bx],bx ; 05/09/2023
000012D2 8CC8                       mov     ax,cs                 ; by overlap jmp instruction of sysinitseg.

                                ;-----------------------------------------------------M067----------------
                                ;
                                ;   the environment pointer is made 0. so the current environment ptr.
                                ;   will be the same as pdb_environ which after dosinit is 0.
                                ;
                                ;   mov     cs:[instexe.exec0_environ],0 ; set the environment seg.
                                ;
                                ;   instexe.exec0_environ need not be initialized to 0 above. It was
                                ;   done as a fix for bug #529. The actual bug was in NLSFUNC and
                                ;   was fixed.
                                ;
                                ; ---------------------------------------------------------------------------

                                ;;ifdef MULTI_CONFIG

                                ; If there's any environment data in "config_wrkseg", pass to app

                                ; 30/12/2022 - Retro DOS v4.0 (Modified MSDOS 6.21 IO.SYS SYSINIT)
                                ; 24/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS SYSINIT)
                                ;%if 0
000012D4 89C1                       mov     cx,ax ; *
                                    ; 05/09/2023
000012D6 2E391E[6019]               cmp     [cs:config_envlen],bx ; 0
                                    ;cmp    word [cs:config_envlen],0
000012DB 7405                       je      short no_envdata2
000012DD 2E8B0E[6219]               mov     cx,[cs:config_wrkseg] ; *
                                no_envdata2:
                                ;;endif  ;MULTI_CONFIG

                                ;%endif     ; 24/10/2022

                                    ;mov    [cs:instexe.exec0_environ],cx ; set the environment seg.
                                    ; 05/09/2023 (BugFix)
                                    ; 24/10/2022
000012E2 2E890E[4203]               mov     [cs:iexec.environ],cx ; *
                                    ; 02/11/2022
                                    ;mov    [cs:iexec.environ],ax ; 05/09/2023

                                    ;mov    [cs:instexe.exec0_com_line+2],ax ; set the seg.
000012E7 2EA3[4603]                 mov     [cs:iexec.ldexec_line+2],ax
                                    ;mov    [cs:instexe.exec0_5c_fcb+2],ax
000012EB 2EA3[4A03]                 mov     [cs:iexec.ldexec_5c_fcb+2],ax
                                    ;mov    [cs:instexe.exec0_6c_fcb+2],ax
000012EF 2EA3[4E03]                 mov     [cs:iexec.ldexec_6c_fcb+2],ax
000012F3 E86000                     call    sum_up
000012F6 26A3[DA02]                 mov     [es:checksum],ax      ; save the value of the sum
000012FA 31C0                       xor     ax,ax
000012FC B44B                       mov     ah,4Bh ; EXEC         ; load/exec
000012FE BB[4203]                   mov     bx,instexe            ; es:bx -> parm block.
00001301 06                         push    es                    ; save es,ds for load/exec
00001302 1E                         push    ds                    ; these registers will be restored in sysinit_base.
00001303 2EFF2E[D202]               jmp     far [cs:sysinit_base_ptr] ; jmp to sysinit_base to execute
                                                                  ; load/exec function and check sum.

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

                                ;j.k. this is the returning address from sysinit_base.

                                    ; 24/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS SYSINIT)

                                sysinitptr:                       ; returning far address from sysinit_base
00001308 5E                         pop     si                    ; restore si for config.sys file.
00001309 06                         push    es
0000130A 1E                         push    ds
0000130B 07                         pop     es
0000130C 1F                         pop     ds                    ; now ds - sysinitseg, es - confbot
0000130D 7305                           jnc     short install_exit_ret

0000130F 56                         push    si                    ; error in loading the file for install=.
00001310 E81937                     call    badload               ; es:si-> path,filename,0.
00001313 5E                         pop     si

                                    ; 24/10/2022
                                    ;jmp    short sysinitptr_retn ; (MSDOS 5.0 IO.SYS, SYSINIT:1140h)
                                    ; 11/12/2022
                                    ; ds = cs

                                    ; 30/12/2022 - Retro DOS v4.2
                                    ; (MSDOS 6.21 IO.SYS, SYSINIT:1283h)

                                install_exit_ret:
00001314 C3                         retn

                                ; 30/12/2022 - Retro DOS v4.2
                                %if 0
                                install_exit_ret:
                                    ;retn          ; retn (MSDOS 6.21 IO.SYS, SYSINIT:1283h) ; 18/12/2022

                                    ; 24/10/2022 (MSDOS 5.0 IO.SYS SYSINIT)
                                ;SYSINIT:1142h:
                                    mov     ah,4Dh
                                    int     21h             ; DOS - 2+ - GET EXIT CODE OF SUBPROGRAM (WAIT)
                                    cmp     ah,3
                                    jz      short sysinitptr_retn
                                    call    error_line
                                    stc
                                sysinitptr_retn:   ; (SYSINIT:114Fh)
                                    retn

                                %endif ; 24/10/2022

                                ; ---------------------------------------------------------------------------
```

```
28006
28007                                   ;** ParaRound - Round Up length to paragraph multiple
28008                                   ;
28009                                   ;   ParaRound rounds a byte count up to a multiple of 16, then divides
28010                                   ;   by 16 yielding a "length in paragraphs" value.
28011                                   ;
28012                                   ;   ENTRY   (ax) = byte length
28013                                   ;   EXIT    (ax) = rounded up length in paragraphs
28014                                   ;   USES    ax, flags
28015
28016                                   ParaRound:
28017 00001315 83C00F                      add     ax,15
28018 00001318 D1D8                        rcr     ax,1
28019 0000131A D1E8                        shr     ax,1
28020 0000131C D1E8                        shr     ax,1
28021 0000131E D1E8                        shr     ax,1
28022 00001320 C3                          retn
28023
28024                                   ; ----------------------------------------------------------------------
28025                                   ; sysinit_base module.
28026                                   ;
28027                                   ; This module is relocated by the routine EndFile to a location in low
28028                                   ; memory. It is then called by SYSINIT to perform the EXEC of programs
28029                                   ; that are being loaded by the "install=" command. After the EXEC call
28030                                   ; completes, this module performs a checksum on the SYSINIT code (at the
28031                                   ; top of memory) to be sure that the EXECed program did not damage it.
28032                                   ; If it did, then this module will print an error message and stop the
28033                                   ; system. Otherwise, it returns control to SYSINIT.
28034                                   ;
28035                                   ;in: after relocation,
28036                                   ;    ax = 4b00h - load and execute the program dos function.
28037                                   ;    ds = confbot. segment of config.sys file image
28038                                   ;    es = sysinitseg. segment of sysinit module itself.
28039                                   ;    ds:dx = pointer to asciiz string of the path,filename to be executed.
28040                                   ;    es:bx = pointer to a parameter block for load.
28041                                   ;    SI_end (byte) - offset vaule of end of sysinit module label
28042                                   ;    bigsize (word) - # of word from confbot to SI_end.
28043                                   ;    chksum (word) - sum of every byte from confbot to SI_end in a
28044                                   ;                    word boundary moduler form.
28045                                   ;    sysinit_ptr (dword ptr) - return address to sysinit module.
28046                                   ;
28047                                   ;note: sysinit should save necessary registers and when the control is back
28048
28049                                       ; 24/10/2022
28050                                       ; (SYSINIT:115Ch for MSDOS 5.0 SYSINIT)
28051                                   sysinit_base:
28052 00001321 2E8C166200                  mov     [cs:sysinit_base_ss],ss       ; save stack
28053 00001326 2E89266400                  mov     [cs:sysinit_base_sp],sp
28054 0000132B CD21                        int     21h                           ; load/exec dos call.
28055 0000132D 2E8E166200                  mov     ss,[cs:sysinit_base_ss]       ; restore stack
28056 00001332 2E8B266400                  mov     sp,[cs:sysinit_base_sp]
28057 00001337 1F                          pop     ds                            ; restore confbot seg
28058 00001338 07                          pop     es                            ; restore sysinitseg
28059 00001339 7216                        jc      short sysinit_base_end        ; load/exec function failed.
28060                                                                            ; at this time,i don't have to worry about
28061                                                                            ; that sysinit module has been broken or not.
28062 0000133B E81800                      call    sum_up                        ; otherwise,check if it is good.
28063 0000133E 263906[DA02]                cmp     [es:checksum],ax
28064 00001343 740C                        je      short sysinit_base_end
28065
28066                                   ;   memory broken. show "memory allocation error" message and stall.
28067
28068 00001345 B409                        mov     ah,9
28069 00001347 0E                          push    cs
28070 00001348 1F                          pop     ds
28071                                       ; 30/12/2022
28072                                       ; (MSDOS 6.21 IO.SYS, SYSINIT:12B8h)
28073                                       ;mov    dx,102
28074 00001349 BA6600                      mov     dx,mem_alloc_err_msgx-sysinit_base ; 65h (for MSDOS 5.0 SYSINIT)
28075                                                                            ; 66h (for MSDOS 6.21 SYSINIT)
28076 0000134C CD21                        int     21h
28077                                              ; DOS - PRINT STRING
28078                                              ; DS:DX -> string terminated by "$"
28079
28080                                       ; 30/12/2022 - Retro DOS v4.2
28081                                   stall:
28082                                       ; 24/10/2022
28083                                   _stall:
28084                                       ; 11/12/2022
28085 0000134E F4                          hlt
28086                                       ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
28087                                       ;hlt                           ;use HLT to minimize energy consumption
28088 0000134F EBFD                            jmp short _stall
28089
28090                                   sysinit_base_end:
28091 00001351 26FF2E[D602]                jmp     far [es:sysinit_ptr]  ;return back to sysinit module
28092
28093                                   ;--------------------------------------
28094
28095                                   sum_up:
28096
28097                                   ;in:   es - sysinitseg.
28098                                   ;out:  ax - result
28099                                   ;
28100                                   ;remark: since this routine will only check starting from "locstack" to the end of
28101                                   ;    sysinit segment,the data area, and the current stack area are not
28102                                   ;    coverd. in this sense,this check sum routine only gives a minimal
28103                                   ;    gaurantee to be safe.
28104                                   ;
28105                                   ;first sum up confbot seg.
28106
28107 00001356 1E                            push    ds
28108                                       ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
28109                                       ; (SYSINIT:12C6h)
28110 00001357 26A1[A302]                  mov     ax,[es:CONFBOT]
28111                                       ; 24/10/2022
28112                                       ;mov    ax,[es:top_of_cdss]
28113 0000135B 8ED8                        mov     ds,ax
28114 0000135D 31F6                        xor     si,si
28115 0000135F 31C0                        xor     ax,ax
28116 00001361 268B0E[D002]                mov     cx,[es:config_size]   ; if config_size has been broken,then this
28117                                                                    ; whole test better fail.
28118 00001366 D1E9                        shr     cx,1                  ; make it a word count
28119 00001368 7406                        jz      short sum_sys_code    ; when config.sys file not exist.
28120                                   sum1:
28121 0000136A 0304                        add     ax,[si]
28122 0000136C 46                          inc     si
28123 0000136D 46                          inc     si
28124 0000136E E2FA                        loop    sum1
28125                                   ;now,sum up sysinit module.
28126                                   sum_sys_code:
28127                                       ; 24/10/2022
28128 00001370 BE7013                      mov     si,locstack ;; 5A6h (MSDOS 5.0 IO.SYS, SYSINIT)
28129                                                          ;; 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:
000013EE 2E0106[6203]               add     [cs:memlo],ax
                                    ;or     byte [cs:setdevmarkflag],for_devmark ; 2
000013F3 2E800E[6919]02             or      byte [cs:setdevmarkflag],2
                                    ;call   round
                                    ;retn
                                    ; 12/12/2022
000013F9 E9F633                     jmp     round

                                ; ----------------------------------------------------------------------
                                ; procedure : GetBufferAddr
                                ;
                                ;         Gets the buffer address either in HMA or in Lo Mem
                                ;
                                ; returns in es:di the buffer adress
                                ; returns NZ if allocated in HMA
                                ; ----------------------------------------------------------------------

                                    ; 25/10/2022
                                GetBufferAddr:
000013FC 53                         push    bx
000013FD 52                         push    dx

                                    ; 11/04/2024 - Retro DOS v5.0
                                    ; PCDOS 7.1 IBMBIO.COM
                                    ;;;
000013FE 2E803E[6E03]02             cmp     byte [cs:dosdata_umb],2
                                                    ; is dosdata moved to UMB ? (DOSDATA=UMB done)
00001404 7506                       jne     short gba_1     ; no
00001406 837F02FF                   cmp     word [bx+2],0FFFFh ; is the buffer (already) in HMA ?
0000140A 7423                       je      short gba_2    ; yes
                                gba_1:
                                    ;;;

0000140C 2EA1[9D02]                 mov     ax, [cs:singlebuffersize]
00001410 2EF726[9902]               mul     word [cs:buffers]
                                    ;add    ax,0Fh
00001415 83C00F                     add     ax,15
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;and    ax,~15  ; 0FFF0h        ; para round
                                    ; 12/12/2022
00001418 24F0                       and     al,~15  ; 0F0h
0000141A 89C3                       mov     bx,ax
0000141C B8024A                     mov     ax,4A02h
                                    ;mov    ax,((multMULT<<8)+multMULTALLOCHMA)
0000141F CD2F                       int     2Fh     ; DOS 5+ - ALLOCATE HMA SPACE
                                                    ;     AX = 4A02h
                                                    ;     BX = number of bytes
                                                    ; Return:
                                                    ;     ES:DI -> start of allocated HMA block or FFFFh:FFFFh
                                                    ;     BX = number of bytes actually allocated
                                                    ;          (rounded up to next paragraph)
                                                    ; Notes:
                                                    ;     this call is not valid unless DOS is loaded in the HMA
                                                    ;     (DOS=HIGH)

00001421 83FFFF                     cmp     di,0FFFFh
00001424 7506                       jne     short got_hma

                                    ;mov    di,0            ; dont xor di,di Z flag needed
                                    ; 05/09/2023
                                    ; zf=1
00001426 47                         inc     di ; 0FFFFh -> 0
                                    ; zf=1

                                    ;zf=1
                                    ;xor    di,di   ; 25/10/2022
                                    ;zf=1
00001427 2E8E06[6403]               mov     es,[cs:memhi]
                                got_hma:
0000142C 5A                         pop     dx
0000142D 5B                         pop     bx
0000142E C3                         retn

                                    ; 11/04/2024 - Retro DOS v5.0
                                    ; PCDOS 7.1 IBMBIO.COM
                                    ;;;
                                gba_2:
0000142F C43F                       les     di,[bx]
00001431 09FF                       or      di,di
                                    ;pop    dx
                                    ;pop    bx
                                    ;retn
                                    ; 11/04/2024 - Retro DOS v5.0
00001433 EBF7                       jmp     short got_hma
                                    ;;;

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

                                set_buffer_info:

                                ;function: set buf_link,buf_id,buf_sector
                                ;
                                ;in: es:di -> buffer header to be set.
                                ;    ax = di
                                ;
                                ;out:
                                ;    above entries set.

                                    ; 25/10/2022
00001435 2EFF36[BD02]               push    word [cs:buf_prev_off]
                                    ;pop    word [es:di+buffinfo.buf_prev]
0000143A 268F4502                   pop     word [es:di+2]
0000143E 2EA3[BD02]                 mov     [cs:buf_prev_off],ax
00001442 2E0306[9D02]               add     ax,[cs:singlebuffersize]      ;adjust ax
                                    ;mov    [es:di+buffinfo.buf_next],ax
00001447 268905                     mov     [es:di],ax
                                    ;mov    word [es:di+buffinfo.buf_ID],00FFh  ; new buffer free
0000144A 26C74504FF00               mov     word [es:di+4],00FFh
                                    ;mov    word [es:di+buffinfo.buf_sector],0   ; to compensate the masm 3 bug
00001450 26C745060000               mov     word [es:di+6],0
                                    ;mov    word [es:di+buffinfo.buf_sector+2],0 ; to compensate the masm 3 bug
00001456 26C745080000               mov     word [es:di+8],0
0000145C C3                         retn

                                ; ==========================================================================
                                ; MSSTACK initialization routine - MSDOS 6.0 - SYSDINIT1.ASM - 1991
                                ; ----------------------------------------------------------------------
                                ; 27/03/2019 - Retro DOS v4.0

                                ; ----------------------------------------------------------------------
                                ; ibmstack initialization routine.
                                ;
                                ;   to follow the standard interrupt sharing scheme, msstack.asm
```

```
28376                                   ;   has been modified. this initialization routine also has to
28377                                   ;   be modified because for the interrupt level 7 and 15, firstflag
28378                                   ;   should be set to signal that this interrupt handler is the
28379                                   ;   first handler hooked to this interrupt vector.
28380                                   ;   we determine this by looking at the instruction pointed by
28381                                   ;   this vector. if it is iret, then this handler should be the
28382                                   ;   first one. in our case, only the interrupt vector 77h is the
28383                                   ;   interrupt level 15. (we don't hook interrupt level 7.)
28384                                   ;
28385                                   ;   the followings are mainly due to m.r.t; ptm fix of p886 12/3/86
28386                                   ;   some design changes are needed to the above interrupt sharing
28387                                   ;   method. the above sharing scheme assumes that 1). interrupt
28388                                   ;   sharing is never done on levels that have bios support. 2). "phantom"
28389                                   ;   interrupts would only be generated on levels 7 and 15.
28390                                   ;   these assumptions are not true any more. we have to use the firstflag
28391                                   ;   for every level of interrupt. we will set the firstflag on the following
28392                                   ;   conditions:
28393                                   ;
28394                                   ;    a.      if the cs portion of the vector is 0000, then "first"
28395                                   ;    b. else if cs:ip points to valid shared header, then not "first"
28396                                   ;    c. else if cs:ip points to an iret, then "first"
28397                                   ;    d. else if cs:ip points to dummy, then "first"
28398                                   ;
28399                                   ;   where dummy is - the cs portion must be f000, and the ip portion must
28400                                   ;   be equal to the value at f000:ff01. this location is the initial value
28401                                   ;   from vector_table for interrupt 7, one of the preserved addresses in all
28402                                   ;   the bioses for all of the machines.
28403                                   ;
28404                                   ;   system design group requests bios to handle the phantom interrupts.
28405                                   ;
28406                                   ;   the "phantom" interrupt is an illegal interrupt such as an interrupt
28407                                   ;   produced by the bogus adapter card even without interrupt request is
28408                                   ;   set.  more specifically, 1). the 8259 has a feature when running in
28409                                   ;   edge triggered mode to latch a pulse and present the interrupt when
28410                                   ;   the processor indicates interrupt acknowledge (inta). the interrupt
28411                                   ;   pulse was exist at the time of inta to get a "phantom" interrupt.
28412                                   ;   2). or, this is caused by adapter cards placing a glitch on the
28413                                   ;   interrupt line.
28414                                   ;
28415                                   ;   to handle those "phantom" interrupts, the main stack code will check
28416                                   ;   the own firstflag, and if it is not "first" (which means the forward
28417                                   ;   pointer points to the legal shared interrupt handler), then pass the
28418                                   ;   control. if it is the first, then the following action should be
28419                                   ;   taken. we don't have to implement skack logic in this case.
28420                                   ;
28421                                   ;   to implement this logic, we rather choose a simple method.
28422                                   ;   if ont of the above "firstflag" conditions is met, we are not
28423                                   ;   going to hook this interrupt vector. the reason is if the original
28424                                   ;   vector points to "iret" and do nothing, we don't need
28425                                   ;   to implement the stack logic for it. this will simplify implementation
28426                                   ;   while maintaining compatibility with the old version of dos.
28427                                   ;   this implies that in the main stack code, there might be a stack code
28428                                   ;   that will never be used, a dead code.
28429                                   ;
28430                                   ;in - cs, ds -> sysinitseg, es -> relocated stack code & data.
28431
28432                                       ; 25/10/2022 - Retro DOS 4.0 (Modified MSDOS 5.0 IO.SYS, SYSINIT)
28433                                       ; (SYSINIT:1287h)
28434
28435                                       ; 11/04/2024 - Retro DOS 5.0 (Modified PCDOS 7.1 IBMBIO.COM)
28436                                       ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:157Ch)
28437
28438                                       ; 14/12/2022
28439                                   stackinit:
28440 0000145D 50                         push  ax
28441 0000145E 1E                         push  ds
28442 0000145F 06                         push  es
28443 00001460 53                         push  bx
28444 00001461 51                         push  cx
28445 00001462 52                         push  dx
28446 00001463 57                         push  di
28447 00001464 56                         push  si
28448 00001465 55                         push  bp
28449
28450                                   ;currently es -> stack code area
28451
28452                                       ; 12/12/2022
28453                                       ; ds = cs
28454 00001466 A1[8C02]                   mov   ax,[stack_count]
28455 00001469 89C1                       mov   cx,ax  ; *!*!*
28456                                     ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
28457                                     ; (MSDOS 5.0 IO.SYS - SYSINIT:1290h)
28458                                     ;mov   ax,[cs:stack_count] ; !!     ;defined in cs
28459 0000146B 26A3[0200]                 mov   [es:stackcount],ax          ;defined in stack code area
28460                                     ; (MSDOS 5.0 IO.SYS - SYSINIT:1298h)
28461 0000146F A1[8E02]                   mov   ax,[stack_size]       ; !!          ;in cs
28462 00001472 26A3[0600]                 mov   [es:stacksize],ax
28463                                     ; 12/12/2022
28464 00001476 A1[9002]                   mov   ax,[stack_addr]                     ; offset
28465                                     ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
28466                                     ; (MSDOS 5.0 IO.SYS - SYSINIT:129Fh)
28467                                     ;mov   ax,[cs:stack_addr]  ; !!
28468 00001479 26A3[0800]                 mov   [es:stacks],ax
28469                                     ; 12/12/2022
28470 0000147D 89C5                       mov   bp,ax ; *!*
28471 0000147F A1[9202]                   mov   ax,[stack_addr+2]
28472                                     ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
28473                                     ; (MSDOS 5.0 IO.SYS - SYSINIT:129Fh)
28474                                     ;mov   ax,[cs:stack_addr+2] ; !!    ; segment
28475 00001482 26A3[0A00]                 mov   [es:stacks+2],ax
28476
28477                                   ; initialize the data fields with the parameters
28478
28479                                   ; "firstentry" will always be at stacks
28480
28481                                       ;mov   bp,[es:stacks]               ; get offset of stack
28482                                       ; 12/12/2022
28483                                       ; bp = [es:stacks] ; *!*
28484 00001486 26892E[0C00]               mov   [es:firstentry],bp
28485
28486                                   ; the stacks will always immediately follow the table entries
28487
28488 0000148B B80800                      mov   ax,entrysize ; 8
28489                                     ;mov   cx,[es:stackcount]
28490                                     ; 12/12/2022
28491                                     ; cx = [es:stackcount] ; *!*!*
28492 0000148E F7E1                       mul   cx
28493 00001490 01E8                       add   ax,bp
28494 00001492 26A3[0400]                 mov   [es:stackat],ax
28495 00001496 89C3                       mov   bx,ax
28496 00001498 83EB02                     sub   bx,2
28497
28498                                   ; zero the entire stack area to start with
28499
```

```
28500 0000149B 268B3E[0400]               mov     di,[es:stackat]
28501 000014A0 26A1[0600]                 mov     ax,[es:stacksize]
28502 000014A4 F7E1                       mul     cx
28503 000014A6 89C1                       mov     cx,ax
28504 000014A8 31C0                       xor     ax,ax
28505 000014AA 06                         push    es
28506 000014AB 1F                         pop     ds                              ;ds = relocated stack code seg.
28507
28508                                   ;now, ds -> stack code area
28509
28510 000014AC 8E06[0A00]                 mov     es,[stacks+2]                   ; get segment of stack area.
28511 000014B0 FC                         cld
28512 000014B1 F3AA                       rep     stosb
28513
28514 000014B3 8B0E[0200]                 mov     cx,[stackcount]
28515
28516                                   ; loop for "count" times, building a table entry
28517                                   ;  cs = sysinitseg, ds = relocated stack code seg, es = segment of stack space
28518                                   ;  cx = number of entries
28519                                   ;  es:bp => base of stacks - 2
28520                                   ;  es:bx => first table entry
28521
28522                                   buildloop:
28523                                       ; 11/12/2022
28524                                       ;mov   byte [es:bp+allocbyte],free   ; mov [es:bp+0],0
28525                                       ; 25/10/2022
28526                                       ;mov   byte [es:bp],free
28527                                       ; 06/07/2023
28528 000014B7 26884600                   mov     [es:bp],al ; 0 ; free
28529 000014BB 26884601                   mov     [es:bp+intlevel],al   ; ax = 0
28530                                     ;mov    [es:bp+1],al
28531 000014BF 26894602                   mov     [es:bp+savedsp],ax
28532                                     ;mov    [es:bp2],ax
28533 000014C3 26894604                   mov     [es:bp+savedss],ax
28534                                     ;mov    [es:bp+4],ax
28535 000014C7 031E[0600]                 add     bx,[stacksize]
28536 000014CB 26895E06                   mov     [es:bp+newsp],bx                ; mov [es:bp+6],bx
28537                                     ;mov    [es:bp+6],bx
28538 000014CF 26892F                     mov     [es:bx],bp
28539 000014D2 83C508                     add     bp,entrysize ; 8
28540
28541 000014D5 E2E0                       loop    buildloop
28542
28543 000014D7 83ED08                     sub     bp,entrysize ; 8
28544 000014DA 892E[0E00]                 mov     [lastentry],bp
28545 000014DE 892E[1000]                 mov     [nextentry],bp
28546
28547 000014E2 1E                          push    ds
28548                                     ;mov    ax,0F000h               ;look at the model byte
28549                                     ; 05/09/2023
28550 000014E3 B4F0                       mov     ah,0F0h ; ax = 0F000h
28551 000014E5 8ED8                       mov     ds,ax
28552 000014E7 803EFEFFF9                 cmp     byte [0FFFEh],0F9h ; mdl_convert ; convertible?
28553 000014EC 1F                         pop     ds
28554 000014ED 7504                       jne     short skip_disablenmis
28555
28556 000014EF B007                       mov     al,07h                  ; disable convertible nmis
28557 000014F1 E672                       out     72h,al
28558
28559                                   skip_disablenmis:
28560 000014F3 31C0                       xor     ax,ax
28561 000014F5 8EC0                       mov     es,ax                   ;es - segid of vector table at 0
28562                                                                     ;ds - relocated stack code segment
28563 000014F7 FA                         cli
28564
28565                                     ;irp    aa,<02,08,09,70>
28566                                     ;
28567                                     ;mov    si,aa&h*4               ;pass where vector is to be adjusted
28568                                     ;mov    di,offset int19old&aa ;we have to set old&aa for int19 handler too.
28569                                     ;mov    bx,offset old&aa        ;pass where to save original owner pointer
28570                                     ;mov    dx,offset int&aa        ;pass where new handler is
28571                                     ;call   new_init_loop           ;adjust the vector to new handler,
28572                                     ;                               ;saving pointer to original owner
28573                                     ;endm
28574
28575                                   stkinit_02:
28576 000014F8 BE0800                     mov     si,02h*4 ; 8
28577 000014FB BF[B305]                   mov     di,INT19OLD02
28578 000014FE BB[1200]                   mov     bx,old02
28579 00001501 BA[1600]                   mov     dx,int02
28580 00001504 E84801                     call    new_init_loop
28581                                   stkinit_08:
28582 00001507 BE2000                     mov     si,08h*4 ; 32
28583 0000150A BF[B805]                   mov     di,INT19OLD08
28584 0000150D BB[3800]                   mov     bx,old08
28585 00001510 BA[3C00]                   mov     dx,int08
28586 00001513 E83901                     call    new_init_loop
28587                                   stkinit_09:
28588 00001516 BE2400                     mov     si,09h*4 ; 36
28589 00001519 BF[BD05]                   mov     di,INT19OLD09
28590 0000151C BB[4100]                   mov     bx,old09
28591 0000151F BA[4500]                   mov     dx,int09
28592 00001522 E82A01                     call    new_init_loop
28593                                   stkinit_70:
28594 00001525 BEC001                     mov     si,70h*4 ; 448
28595 00001528 BF[DB05]                   mov     di,INT19OLD70
28596 0000152B BB[4E00]                   mov     bx,old70
28597 0000152E BA[5200]                   mov     dx,int70
28598 00001531 E81B01                     call    new_init_loop
28599
28600                                     ;irp    aa,<0a,0b,0c,0d,0e,72,73,74,76,77> ;shared interrupts
28601                                     ;
28602                                     ;mov    si,aa&h*4               ;pass where vector is to be adjusted
28603                                     ;push   ds                      ;save relocated stack code segment
28604                                     ;lds    bx, es:[si]             ;ds:bx -> original interrupt handler
28605                                     ;push   ds
28606                                     ;pop    dx                      ;dx = segment value
28607                                     ;
28608                                     ;cmp    dx,0
28609                                     ;jz     int&aa&_first
28610                                     ;
28611                                     ;cmp    byte ptr ds:[bx],0cfh ;does vector point to an iret?
28612                                     ;jz     int&aa&_first
28613                                     ;
28614                                     ;cmp    word ptr ds:[bx.6],424bh ;magic offset (see int&aa, msstack.inc)
28615                                     ;jz     int&aa&_not_first
28616                                     ;
28617                                     ;cmp    dx,0f000h               ;rom bios segment
28618                                     ;jnz    int&aa&_not_first
28619                                     ;
28620                                     ;push   es
28621                                     ;push   dx
28622                                     ;mov    dx,0f000h
28623                                     ;mov    es,dx
```

```
                                    ;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
```

```
28872 0000158A 730C                     jnc     short int_0E_end ; int_0E_first
28873
28874                                   ; 14/12/2022
28875                                   %if 0
28876                                       ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
28877                                       push    ds
28878                                       lds     bx,[es:si]
28879                                       push    ds
28880                                       pop     dx
28881
28882                                       cmp     dx,0
28883                                       je      short int_0E_first
28884
28885                                       cmp     byte [bx],0CFh
28886                                       je      short int_0E_first
28887
28888                                       cmp     word [bx+6],424Bh
28889                                       je      short int_0E_not_first
28890
28891                                       cmp     dx,0F000h
28892                                       jne     short int_0E_not_first
28893
28894                                       push    es
28895                                       push    dx
28896                                       mov     dx,0F000h
28897                                       mov     es,dx
28898                                       cmp     bx,[es:0FFF1h]
28899                                       pop     dx
28900                                       pop     es
28901                                       je      short int_0E_first
28902                                   %endif
28903
28904                                   int_0E_not_first:
28905                                       ; 14/12/2022
28906                                       ; 25/10/2022
28907                                       ;pop    ds
28908 0000158C BF[D605]                   mov     di,INT19OLD0E
28909 0000158F BB[B900]                   mov     bx,old0E
28910 00001592 BA[B700]                   mov     dx,int0E
28911 00001595 E8B700                     call    new_init_loop
28912
28913                                       ; 14/12/2022
28914                                       ;jmp    short int_0E_end
28915                                   ;int_0E_first:
28916                                       ; 25/10/2022
28917                                       ;pop    ds
28918
28919                                   int_0E_end:
28920
28921                                   stkinit_72:
28922 00001598 BEC801                     mov     si,72h*4 ; 456
28923
28924                                       ; 14/12/2022
28925                                       ; 25/10/2022
28926 0000159B E88700                     call    int_xx_first_check
28927 0000159E 730C                       jnc     short int_72_end ; int_72_first
28928
28929                                   ; 14/12/2022
28930                                   %if 0
28931                                       ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
28932                                       push    ds
28933                                       lds     bx,[es:si]
28934                                       push    ds
28935                                       pop     dx
28936
28937                                       cmp     dx,0
28938                                       je      short int_72_first
28939
28940                                       cmp     byte [bx],0CFh
28941                                       je      short int_72_first
28942
28943                                       cmp     word [bx+6],424Bh
28944                                       je      short int_72_not_first
28945
28946                                       cmp     dx,0F000h
28947                                       jne     short int_72_not_first
28948
28949                                       push    es
28950                                       push    dx
28951                                       mov     dx,0F000h
28952                                       mov     es,dx
28953                                       cmp     bx,[es:0FFF1h]
28954                                       pop     dx
28955                                       pop     es
28956                                       je      short int_72_first
28957                                   %endif
28958
28959                                   int_72_not_first:
28960                                       ; 14/12/2022
28961                                       ; 25/10/2022
28962                                       ;pop    ds
28963 000015A0 BF[E005]                   mov     di,INT19OLD72
28964 000015A3 BB[D100]                   mov     bx,old72
28965 000015A6 BA[CF00]                   mov     dx,int72
28966 000015A9 E8A300                     call    new_init_loop
28967
28968                                       ; 14/12/2022
28969                                       ;jmp    short int_72_end
28970                                   ;int_72_first:
28971                                       ; 25/10/2022
28972                                       ;pop    ds
28973
28974                                   int_72_end:
28975
28976                                   stkinit_73:
28977 000015AC BECC01                     mov     si,73h*4 ; 460
28978
28979                                       ; 14/12/2022
28980                                       ; 25/10/2022
28981 000015AF E87300                     call    int_xx_first_check
28982 000015B2 730C                       jnc     short int_73_end ; int_73_first
28983
28984                                   ; 14/12/2022
28985                                   %if 0
28986                                       ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
28987                                       push    ds
28988                                       lds     bx,[es:si]
28989                                       push    ds
28990                                       pop     dx
28991
28992                                       cmp     dx,0
28993                                       je      short int_73_first
28994
28995                                       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
000015B4 BF[E505]                   ;pop    ds
                                    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
000015C8 BF[EA05]                   ;pop    ds
                                    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
00001602 BF[F405]                   mov     di,INT19OLD77
00001605 BB[3101]                   mov     bx,old77
00001608 BA[2F01]                   mov     dx,int77
0000160B 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
0000166A 1E                         push    ds
0000166B 268B4402                   mov     ax,[es:si+2]            ;remember segid in vector
0000166F 894702                     mov     [bx+2],ax               ; to original owner in ds
00001672 50                         push    ax
00001673 268B04                     mov     ax,[es:si]              ;remember offset in vector
00001676 8907                       mov     [bx],ax                 ; to original owner in ds
00001678 50                         push    ax
00001679 B87000                     mov     ax,DOSBIODATASEG ; 0070h
0000167C 8ED8                       mov     ds,ax                   ;set int19oldxx value in bios for
0000167E 58                         pop     ax                      ;int 19 handler
0000167F 8905                       mov     [di],ax
00001681 58                         pop     ax
00001682 894502                     mov     [di+2],ax
00001685 1F                         pop     ds
                                %endif
00001686 268914                     mov     [es:si],dx      ;set vector to point to new int handler
00001689 268C5C02                   mov     [es:si+2],ds
0000168D 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)
```

```
29368
29369                                   setdevmark:
29370
29371                                       ; 04/09/2023
29372                                       ;push   es
29373                                       ;push   cx
29374
29375 0000168B 2E8B0E[6403]               mov     cx,[cs:memhi]
29376 00001690 2E890E[6719]               mov     [cs:devmark_addr],cx
29377 00001695 8EC1                       mov     es,cx
29378                                     ; 25/10/2022
29379                                     ;mov    [es:devmark.id],al
29380 00001697 26A20000                   mov     [es:0],al
29381 0000169B 41                         inc     cx
29382                                     ;mov    [es:devmark.seg],cx
29383 0000169C 26890E0100                 mov     [es:1],cx
29384
29385                                     ; 04/09/2023
29386                                     ;pop    cx
29387                                     ;pop    es
29388
29389 000016A1 2EFF06[6403]               inc     word [cs:memhi]
29390 000016A6 C3                         retn
29391
29392                                   ; ----------------------------------------------------------------------
29393                                   ; SYSPRE.ASM - MSDOS 6.0 - 1992
29394                                   ; ----------------------------------------------------------------------
29395                                   ;; pre-load and final placement of dblspace.bin
29396                                   ;
29397                                   ; 08/04/2024 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
29398                                   ; ======================================================
29399
29400                                   ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:1964h)
29401                                   ;;; ----------------------------------------------------------------------
29402 000016A7 [AB16]                   MagicDDNamePtr:    dw MagicDDName ; "\DBLSPACE.BIN"
29403 000016A9 433A                               db 'C:'
29404 000016AB 5C44424C5350414345-     MagicDDName:       db '\DBLSPACE.BIN',0
29404 000016B4 2E42494E00
29405 000016B9 433A5C535441434B45-     StackerName:       db 'C:\STACKER.BIN',0
29405 000016C2 522E42494E00
29406                                   tiny_stub_start:
29407 000016C8 FFFF                               dw 0FFFFh      ; phony device driver link
29408 000016CA FFFF                               dw 0FFFFh      ; dw -1, -1
29409 000016CC 0080                               dw 8000h       ; mark as character device for MEM display
29410 000016CE 00000000                           dw 2 dup(0)    ; strategy and interrupt
29411 000016D2 44424C5342494E24                   db 'DBLSBIN$'  ; magic default load
29412                                   tiny_stub_end:                     ; (tiny_stub_end-tiny_stub_start = 18)
29413
29414                                   ; =============== S U B R O U T I N E ===================================
29415
29416                                   ; 08/04/2024 - Retro DOS v5.0
29417                                   ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:1997h)
29418
29419                                   ;***MagicPreload - pre-load dblspace.bin
29420                                   ;
29421                                   ;   EXIT    ax = error code, 00 means none.
29422                                   ;           ZF = true if ax == 0
29423
29424                                   MagicPreload:
29425                                       ; 13/04/2024 - Retro DOS v5.0
29426                                       ; ds = cs
29427                                       ;mov    byte [cs:setdevmarkflag],0 ; not for devmark
29428 000016DA C606[6919]00               mov     byte [setdevmarkflag],0
29429 000016DF E81031                     call    round
29430 000016E2 0E                         push    cs
29431 000016E3 07                         pop     es
29432                                     ;mov    byte [cs:DeviceHi],0  ; not to be loaded in UMB
29433                                     ; 13/04/2024
29434 000016E4 C606[5124]00               mov     byte [DeviceHi],0
29435 000016E9 E8791E                     call    InitDevLoad            ; set up sub-arena, DevLoadAddr,
29436                                                                    ;  DevLoadEnd, and DevEntry
29437                                                                    ;  gets arena name from bpb_addr
29438                                       ; 13/04/2024
29439                                       ; ds = cs
29440
29441                                   ;   check to make sure device driver fits our available space.
29442
29443                                       ;mov    ax,[cs:DevLoadAddr]
29444 000016EC A1[3524]                   mov     ax,[DevLoadAddr]
29445                                     ;add    ax,[cs:DevSize]              ; calculate seg after DD load
29446 000016EF 0306[3324]                 add     ax,[DevSize]
29447 000016F3 725E                       jc      short pre_exit_err     ; choke if overflows address space
29448                                     ;cmp    ax,[cs:DevLoadEnd]     ; does it overflow available space?
29449 000016F5 3B06[3724]                 cmp     ax,[DevLoadEnd]
29450 000016F9 7758                       ja      short pre_exit_err
29451
29452                                   _LoadDev:                          ; we're golden if not
29453                                       ; 13/04/2024
29454                                       ; ds = cs
29455                                       ;push   cs
29456                                       ;pop    ds
29457                                       ;mov    dx,[cs:MagicDDNamePtr]
29458 000016FB 8B16[A716]                 mov     dx,[MagicDDNamePtr]
29459 000016FF E8A41F                     call    ExecDev                ; load device driver using exec call
29460 00001702 724F                       jb      short pre_exit_err
29461
29462                                     ; 13/04/2024
29463                                     ; ds = cs
29464                                     ;les    bx,[cs:DevEntry]       ; point to the Magic DD header
29465 00001704 C41E[3924]                 les     bx,[DevEntry]
29466 00001708 26817F122C2E               cmp     word [es:bx+12h],2E2Ch; is it our stamp? ; ',.'
29467 0000170E 7543                       jnz     short pre_exit_err
29468                                     ;mov    word [cs:MagicBackdoor],14h ; save the backdoor entry.
29469                                     ;                              ; (initial IP -EXE header offset 20-)
29470                                     ;mov    [cs:MagicBackdoor+2],es
29471 00001710 C706[9003]1400             mov     word [MagicBackdoor],14h
29472 00001716 8C06[9203]                 mov     [MagicBackdoor+2],es
29473
29474 0000171A 0E                         push    cs
29475 0000171B 07                         pop     es
29476 0000171C BB[6F03]                   mov     bx,packet
29477
29478                                     ;mov    word [cs:break_addr],0
29479                                     ;mov    ax,[cs:DevLoadEnd]
29480                                     ;mov    [cs:break_addr+2],ax
29481                                     ;mov    al,[cs:drivenumber]    ; pass drive number to DBLSPACE as if
29482                                     ;mov    [cs:devdrivenum],al    ; it is a normal block device driver
29483 0000171F C706[7D03]0000             mov     word [break_addr],0
29484 00001725 A1[3724]                   mov     ax,[DevLoadEnd]
29485 00001728 A3[7F03]                   mov     [break_addr+2],ax
29486 0000172B A0[8503]                   mov     al,[drivenumber]       ; pass drive number to DBLSPACE as if
29487 0000172E A2[8503]                   mov     [devdrivenum],al       ; it is a normal block device driver
29488
29489 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]
```

```
29614 000017EA EBF3                     jmp     short _scandpb
29615
29616                                ;   We've found the end of the DPB chain. Now extend it.
29617
29618                                _foundpb:
29619 000017EC 2EA1[3D24]               mov     ax,[cs:DevBrkAddr]
29620 000017F0 26894619                 mov     [es:bp+19h],ax          ; [es:bp.dpb_next_dpb] ; DPB.NEXT_DPB
29621 000017F4 2EA1[3F24]               mov     ax,[cs:DevBrkAddr+2]
29622 000017F8 2689461B                 mov     [es:bp+1Bh],ax          ; [es:bp.dpb_next_dpb+2] ; DPB.NEXT_DPB+2
29623 000017FC 2EC42E[3D24]             les     bp,[cs:DevBrkAddr]
29624 00001801 26C74619FFFF             mov     word [es:bp+19h],0FFFFh ; -1
29625 00001807 26C64618FF               mov     byte [es:bp+18h],0FFh ; [es:bp.dpb_first_access],-1
29626                                                                   ; DPB.FIRST_ACCESS
29627 0000180C 2E8306[3D24]3D           add     word [cs:DevBrkAddr],61 ; DPBSIZ ; 3Dh
29628 00001812 E8D01E                   call    RoundBreakAddr
29629 00001815 8B37                     mov     si,[bx]                 ; ds:si points to bpb (*)
29630                                                                   ; (mov si,[bx] ..and then.. add bx,2)
29631                                                                   ; Note: If unit count > 1,bx points to a BPB in the BPB array,
29632                                                                   ; the array address is in [bpb_addr] (*)
29633                                                                   ; Erdogan Tan - 07/07/2023
29634 00001817 26885600                 mov     [es:bp+0],dl                    ; mov word [es:bp.dpb_drive],dx
29635                                                                   ; [es:bp+DPB.DRIVE],dl
29636 0000181B 26887601                 mov     [es:bp+1],dh                    ; [es:bp+DPB.UNIT],dh
29637 0000181F 52                       push    dx
29638 00001820 51                       push    cx
29639 00001821 BA5241                   mov     dx,4152h                ; DX = signature 4152h ('AR') for FAT32 extended BPB/DPB
29640 00001824 31C9                     xor     cx,cx   ; 0
29641 00001826 26894E1D                 mov     [es:bp+1Dh],cx          ; DPB.NEXT_FREE ; last allocated cluster #
29642 0000182A 394C0B                   cmp     [si+0Bh],cx             ; BPB.fatsecs16 ; [si+A_BPB.BPB_SECTORSPERFAT]
29643 0000182D 7514                     jnz     short _setdpb           ; FAT DPB (33 bytes)
29644                                                                   ; FAT32 DPB (61 bytes)
29645 0000182F 26894E39                 mov     [es:bp+39h],cx          ; DPB.FAT32_NXTFREE = 0
29646 00001833 26894E3B                 mov     [es:bp+3Bh],cx          ; DPB.FAT32_NXTFREE+2 = 0
29647 00001837 49                       dec     cx                      ; 0FFFFh ; -1
29648 00001838 26894E1F                 mov     [es:bp+1Fh],cx          ; DPB.FREE_CNT (-1 = unknown)
29649 0000183C 26894E21                 mov     [es:bp+21h],cx          ; DPB.FREE_CNT+2 (-1 = unknown)
29650 00001840 B95845                   mov     cx,4558h                ; CX = signature 4558h ('EX') for FAT32 extended BPB/DPB
29651
29652                                _setdpb:
29653 00001843 B453                      mov     ah,53h          ; SETDPB ; hidden system call
29654 00001845 CD21                     int     21h             ; DOS - 2+ internal - TRANSLATE BIOS PARAMETER BLOCK
29655                                                           ; DS:SI -> BPB (BIOS Parameter Block)
29656                                                           ; ES:BP -> buffer for DOS Drive Parameter Block
29657                                                           ; (if CX=4558h & DX=4152h,FAT32 Extended DPB will be set)
29658 00001847 59                       pop     cx
29659 00001848 5A                       pop     dx
29660 00001849 268B4602                 mov     ax,[es:bp+2]            ; [es:bp.dpb_sector_size]  ; [es:bp+DPB.SECTOR_SIZE]
29661 0000184D 06                       push    es
29662 0000184E 2EC43E[6D02]             les     di,[cs:DOSINFO]
29663 00001853 263B4510                 cmp     ax,[es:di+10h]          ; [es:di.sysi_maxsec] ; [es:di+SysInitvars.SYSI_MAXSEC]
29664 00001857 07                       pop     es
29665 00001858 7604                     jbe     short _iblk_1
29666                                   ; 13/04/2024
29667                                   ; ds <> cs
29668 0000185A B84000                   mov     ax,40h                  ; SYSPRE_BADFILE_ERROR ; (pre_exit_err)
29669                                                                   ; (problem loading dblspace.bin)
29670 0000185D C3                       retn
29671
29672                                _iblk_1:
29673 0000185E 1E                       push    ds
29674 0000185F 2EC506[3924]             lds     ax,[cs:DevEntry]
29675 00001864 26894613                 mov     [es:bp+13h],ax          ; [es:bp+DPB.DRIVER_ADDR]
29676 00001868 268C5E15                 mov     [es:bp+15h],ds          ; [es:bp+DPB.DRIVER_ADDR+2]
29677 0000186C 1F                       pop     ds
29678 0000186D FEC2                     inc     dl                      ; increment drive number
29679 0000186F FEC6                     inc     dh                      ; increment unit number
29680 00001871 43                       inc     bx
29681 00001872 43                       inc     bx                      ; point to next BPB
29682                                                                   ; (in the BPB array) (*) -add bx,2-
29683 00001873 49                       dec     cx                      ; loop _foundpb
29684 00001874 7403                     jz      short _linkit
29685 00001876 E973FF                   jmp     _foundpb
29686
29687                                _linkit:
29688 00001879 0E                        push    cs
29689 0000187A 1F                       pop     ds
29690 0000187B E825F5                   call    TempCDS                 ; set cds for new drives
29691                                   ; 13/04/2024
29692                                   ; (DS may not be same with CS here)
29693 0000187E 2EC43E[6D02]             les     di,[cs:DOSINFO]                 ; es:di = dos table (SysInitVars)
29694 00001883 268B4522                 mov     ax,[es:di+22h]          ; [es:di+SYSI_DEV] ; dx:cx = head of list
29695 00001887 268B5D24                 mov     bx,[es:di+24h]          ; [es:di+SYSI_DEV+2]
29696 0000188B 2EC536[3924]             lds     si,[cs:DevEntry]        ; ds:si = device location
29697 00001890 8904                     mov     [si],ax                 ; link in the driver
29698 00001892 895C02                   mov     [si+2],bx
29699 00001895 26897522                 mov     [es:di+22h],si          ; [es:di+SYSI_DEV] ; set head of list in dos
29700 00001899 268C5D24                 mov     [es:di+24h],ds          ; [es:di+SYSI_DEV+2]
29701 0000189D E8881E                   call    DevBreak                ; mark successful install
29702                                   ; 13/04/2024
29703                                   ; ds = cs
29704                                   ;mov    cx,[cs:DevBrkAddr+2]    ; pass it a work buffer
29705                                   ;mov    dx,[cs:ALLOCLIM]        ; address in cx (segment)
29706 000018A0 8B0E[3F24]               mov     cx,[DevBrkAddr+2]
29707 000018A4 8B16[A502]               mov     dx,[ALLOCLIM]
29708 000018A8 29CA                     sub     dx,cx                   ; for len dx (paragraphs)
29709 000018AA B80055                   mov     ax,5500h                ; we're shuffle aware,but don't move
29710                                                                   ; any drives at this point.
29711 000018AD BB0200                   mov     bx,2                    ; switch what we can now
29712                                   ;call   far [cs:MagicBackdoor]
29713 000018B0 FF1E[9003]               call    far [MagicBackdoor]
29714                                pre_exit:                          ; no errors!
29715 000018B4 31C0                     xor     ax,ax                   ; zf=1
29716                                no_magic:   ; 13/04/2024
29717 000018B6 C3                       retn
29718
29719                                ; =============== S U B R O U T I N E ===================================
29720
29721                                ; 08/04/2024 - Retro DOS v5.0
29722                                ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:1B9Fh)
29723
29724                                ;***MagicPostload -- called to clean up and make sure Magic is final placed
29725
29726                                MagicPostload:
29727                                   ; 13/04/3024
29728                                   ; ds = cs
29729 000018B7 E89C00                  call    get_dblspace_version  ; is it there?
29730 000018BA 75FA                     jnz     short no_magic        ; done if not
29731 000018BC F7C20080                 test    dx,8000h              ; is it already permanent?
29732 000018C0 74F4                     jz      short no_magic        ; no,done if so (not in final position)
29733 000018C2 BBFFFF                   mov     bx,0FFFFh ; -1        ; how much space does it want?
29734 000018C5 B8114A                   mov     ax,4A11h              ; multMagicdrv
29735                                                                 ; DBLSPACE.BIN - GET RELOCATION SIZE
29736 000018C8 CD2F                     int     2Fh                   ; get paragraphs into ax
29737 000018CA 40                       inc     ax                    ; extra 2 paragraphs for the stub
```

```
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:
                                        push    cx
                                        push    dx
                                        push    ds
                                        push    si
00001934 B8114A                         mov     ax,4A11h                ; multMagicdrv
00001937 BB0100                         mov     bx,1                    ; MD_DRIVE_MAP ; inquire drive map
0000193A CD2F                           int     2Fh                     ; DBLSPACE.BIN - "GetDriveMapping"
                                                                        ; see if this is an unused DblSpace drive
0000193C 5E                             pop     si
0000193D 1F                             pop     ds
0000193E 5A                             pop     dx
0000193F 59                             pop     cx
00001940 38DA                           cmp     dl,bl                   ; if mapped to itself,it is vacant
00001942 7504                           jnz     short magic_set_cdss_2  ; skip if used
00001944 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:
00001948 83C658                         add     si,88                   ; curdirLen
0000194B FEC2                           inc     dl                      ; next drive
0000194D E2E5                           loop    magic_set_cdss_1
                                        ; 13/04/2024
00001951 0E                             push    cs
00001952 1F                             pop     ds
                                magic_set_exit:
00001953 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
```

```
30110                                   ;           :                       ; REPEATS maxs-1 TIMES
30111                                   ;           DB      maxk            ; # OF KEYWORD
30112                                   ;           DW      CONTROL         ; DESCRIPTION OF KEYWORD 1
30113                                   ;           :                       ; REPEATS maxk-1 TIMES
30114                                   ; ---------------------------------------------------------------------------
30115
30116                                   ;-------------------------------- PARMSX block structure
30117                                   struc _$P_PARMSX_Blk                ;AN000;
30118 00000000 ??                        .MinP: resb 1                      ;AN000; Minimum positional number
30119 00000001 ??                        .MaxP:     resb 1                  ;AN000; Maximum positional number
30120 00000002 ????                      .1st_Control: resw 1               ;AN000; Address of the 1st CONTROL block
30121                                   endstruc
30122
30123                                   ; ---------------------------------------------------------------------------
30124                                   ; << Control field definition  >>
30125                                   ;
30126                                   ;
30127                                   ;CONTROL   LABEL   BYTE
30128                                   ;       DW      MATCH_FLAGS     ; CONTROLS TYPE MATCHED
30129                                   ;                               ; 8000H=NUMERIC VALUE, (VALUE LIST WILL BE CHECKED)
30130                                   ;                               ; 4000H=SIGNED NUMERIC VALUE (VALUE LIST WILL BE CHECKED)
30131                                   ;                               ; 2000H=SIMPLE STRING(VALUE LIST WILL BE CHECKED)
30132                                   ;                               ; 1000H=DATE STRING (VALUE LIST WON'T BE CHECKED)
30133                                   ;                               ; 0800H=TIME STRING (VALUE LIST WON'T BE CHECKED)
30134                                   ;                               ; 0400H=COMPLEX LIST (VALUE LIST WON'T BE CHECKED)
30135                                   ;                               ; 0200H=FILE SPEC (VALUE LIST WON'T BE CHECKED)
30136                                   ;                               ; 0100H=DRIVE ONLY (VALUE LIST WON'T BE CHECKED)
30137                                   ;                               ; 0080H=QUOTED STRING (VALUE LIST WON'T BE CHECKED)
30138                                   ;                               ; 0010H=IGNORE ":" AT END IN MATCH
30139                                   ;                               ; 0002H=REPEATS ALLOWED
30140                                   ;                               ; 0001H=OPTIONAL
30141                                   ;       DW      FUNCTION_FLAGS
30142                                   ;                               ; 0001H=CAP RESULT BY FILE TABLE
30143                                   ;                               ; 0002H=CAP RESULT BY CHAR TABLE
30144                                   ;                               ; 0010H=REMOVE ":" AT END
30145                                   ; (tm10)                        ; 0020H=colon is not necessary for switch
30146                                   ;       DW      RESULT          ; RESULT BUFFER
30147                                   ;       DW      VALUES          ; VALUE LISTS
30148                                   ;       DB      nid             ; NUMBER OF KEYWORD/SWITCH SYNONYMS IN FOLLOWING LIST
30149                                   ;       DB      "...",0          ; IF n >0, KEYWORD 1
30150                                   ;       :
30151                                   ;
30152                                   ;Note:
30153                                   ;     - The MATCH_FLAG is bit significant. You can set, for example, TIME bit and
30154                                   ;       DATE bit simalteniously.
30155                                   ;
30156                                   ;       The parser examins each bit along with the following priority.
30157                                   ;
30158                                   ;       COMPLEX -> DATE -> TIME -> NUMERIC VAL -> SIGNED NUMERIC VAL -> DRIVE ->
30159                                   ;       FILE SPEC -> SIMPLE STRING.
30160                                   ;
30161                                   ;     - When the FUNCTION_FLAG is 0001 or 0002, the STRING pointed to by a pointer
30162                                   ;       in the result buffer is capitalized.
30163                                   ;
30164                                   ;     - Match_Flags 0001H and 0002H have meaning only for the positional.
30165                                   ;
30166                                   ;     - The "...",0 (bottom most line) does require '=' or '/'. When you need a
30167                                   ;       switch, for example, '/A', then STRING points to;
30168                                   ;
30169                                   ;                   DB    1         ; number of following synonyms
30170                                   ;                   DB   '/A',0
30171                                   ;
30172                                   ;       When you need a keyword, for example, 'CODEPAGE=', then "...",0 will be;
30173                                   ;
30174                                   ;                   DB    1         ; number of following synonyms
30175                                   ;                   DB   'CODEPAGE=',0
30176                                   ;
30177                                   ;     - "..." must consist of upper case characters only because the parser
30178                                   ;       performs pattern matching after converting input to upper case (by
30179                                   ;       using the current country upper case table)
30180                                   ;
30181                                   ;     - One "..." can contain only one switch or keyword. If you need, for
30182                                   ;       example /A and /B, the format will be;
30183                                   ;
30184                                   ;                   DB    2         ; number of following synonyms
30185                                   ;                   DB    '/A',0
30186                                   ;                   DB    '/B',0
30187                                   ; ---------------------------------------------------------------------------
30188
30189                                   ;**** Match_Flags
30190
30191                                   _$P_Num_Val    equ 8000h            ;AN000; Numeric Value
30192                                   _$P_SNum_Val       equ 4000h        ;AN000; Signed numeric value
30193                                   _$P_Simple_S       equ 2000h        ;AN000; Simple string
30194                                   _$P_Date_S     equ 1000h            ;AN000; Date string
30195                                   _$P_Time_S     equ 0800h            ;AN000; Time string
30196                                   _$P_Cmpx_S     equ 0400h            ;AN000; Complex string
30197                                   _$P_File_Spc       equ 0200h        ;AN000; File Spec
30198                                   _$P_Drv_Only       equ 0100h        ;AN000; Drive Only
30199                                   _$P_Qu_String      equ 0080h        ;AN000; Quoted string
30200                                   _$P_Ig_Colon       equ 0010h        ;AN000; Ignore colon at end in match
30201                                   _$P_Repeat     equ 0002h        ;AN000; Repeat allowed
30202                                   _$P_Optional       equ 0001h        ;AN000; Optional
30203
30204                                   ;**** Function flags
30205
30206                                   _$P_CAP_File       equ 0001h        ;AN000; CAP result by file table
30207                                   _$P_CAP_Char       equ 0002h        ;AN000; CAP result by character table
30208                                   _$P_Rm_Colon       equ 0010h        ;AN000; Remove ":" at the end
30209                                   _$P_colon_is_not_necessary equ 0020h    ;AN000;(tm10) /+10 and /+:10
30210
30211                                   ;-------------------------------- Control block structure
30212                                   struc _$P_Control_Blk
30213 00000000 ????                      .Match_Flag:       resw 1          ;AN000; Controls type matched
30214 00000002 ????                      .Function_Flag: resw 1             ;AN000; Function should be taken
30215 00000004 ????                      .Result_Buf:       resw 1          ; Result buffer address
30216 00000006 ????                      .Value_List:       resw 1          ;AN000; Value list address
30217 00000008 ??                        .nid:              resb 1          ;AN000; # of keyword/SW synonyms
30218 00000009 ??                        .KEYorSW:   resb 1          ;AN000; keyword or sw
30219                                   endstruc
30220
30221                                   ; ---------------------------------------------------------------------------
30222                                   ; << Value List Definition >>
30223                                   ;
30224                                   ;VALUES     LABEL   BYTE
30225                                   ;           DB      nval                ; NUMBER OF VALUE DEFINITIONS (0 - 3)
30226                                   ;         +-
30227                                   ;         | DB      nrng                ; NUMBER OF RANGES
30228                                   ;         | +DB     ITEM_TAG            ; RETURN VALUE IF RANGE MATCHED
30229                                   ;         | +DD     X,Y                 ; RANGE OF VALUES
30230                                   ;         | :
30231                                   ;         | DB      nnval               ; NUMBER OF CHOICES
30232                                   ;         | +DB     ITEM_TAG            ; RETURN VALUE IF NUMBER CHOICE MATCHED
30233                                   ;         | +DD     VALUE               ; SPECIFIC CHOICE IF NUMBER
```

```
30234                                   ;       |  :
30235                                   ;       |  DB     nstrval             ; NUMBER OF CHOICES
30236                                   ;       |  +DB    ITEM_TAG            ; RETURN VALUE IF STRING CHOICE MATCHED
30237                                   ;       |  +DW    STRING              ; SPECIFIC CHOICE IF STING
30238                                   ;       +- :
30239                                   ;
30240                                   ;STRING    DB     "...",0               ; ASCIIZ STRING IMAGE
30241                                   ;
30242                                   ;Note:
30243                                   ;    - ITEM_TAG must not be 0FFH, which will be used in the result buffer
30244                                   ;      when no choice lists are provided.
30245                                   ;
30246                                   ;    - STRING must consist of upper case characters only because the parser
30247                                   ;      performs pattern matching after converting input to upper case (by
30248                                   ;      using the current country upper case table)
30249                                   ; ---------------------------------------------------------------------------
30250
30251                                   _$P_nval_None         equ 0             ;AN000; no value list ID
30252                                   _$P_nval_Range        equ 1             ;AN000; range list ID
30253                                   _$P_nval_Value        equ 2             ;AN000; value list ID
30254                                   _$P_nval_String       equ 3             ;AN000; string list ID
30255                                   _$P_Len_Range         equ 9             ;AN000; Length of a range choice(two DD plus one DB)
30256                                   _$P_Len_Value         equ 5             ;AN000; Length of a value choice(one DD plus one DB)
30257                                   _$P_Len_String        equ 3             ;AN000; Length of a string choice(one DW plus one DB)
30258                                   _$P_No_nrng    equ 0              ;AN000; (tm07) no nrng. nnval must not be 0.
30259
30260                                   struc _$P_Val_List
30261 00000000 ??                        .NumofList: resb 1                  ;AN000; number of following choice
30262 00000001 ????                      .Val_XL:    resw 1                  ;AN000; lower word of value
30263 00000003 ????                      .Val_XH:    resw 1                  ;AN000; higher word of value
30264 00000005 ????                      .Val_YL:    resw 1                  ;AN000; lower word of another value
30265 00000007 ????                      .Val_YH:    resw 1                  ;AN000; higher word of another value
30266                                   endstruc
30267
30268                                   ; ---------------------------------------------------------------------------
30269                                   ; << Result Buffer Definition  >>
30270                                   ;
30271                                   ;RESULT    LABEL   BYTE                ; BELOW FILLED IN FOR DEFAULTS
30272                                   ;          DB      type                ; TYPE RETURNED: 0=RESERVED,
30273                                   ;                                      ;       1=NUMBER, 2=LIST INDEX,
30274                                   ;                                      ;       3=STRING, 4=COMPLEX,
30275                                   ;                                      ;       5=FILESPEC, 6=DRIVE
30276                                   ;                                      ;       7=DATE, 8=TIME
30277                                   ;                                      ;       9=QUOTED STRING
30278                                   ;          DB      ITEM_TAG            ; MATCHED ITEM TAG
30279                                   ;
30280                                   ;          dw      synonym@            ; es:@ points to found SYNONYM if provided.
30281                                   ;
30282                                   ;            +-
30283                                   ;       | DD      n                   ; VALUE IF NUMBER
30284                                   ;       | or
30285                                   ;       |  DW     i                   ; INDEX (OFFSET) INTO VALUE LIST
30286                                   ;       |                             ; (ES presents Segment address)
30287                                   ;       | or
30288                                   ;       |  DD     STRING              ; OFFSET OF STRING VALUE
30289                                   ;       | or
30290                                   ;       |  DB     drv                 ; DRIVE NUMBER (1-A, 2-B,..., 26-Z)
30291                                   ;       | or
30292                                   ;       |  DW     YEAR      ;(1980-2099)  IN CASE OF DATE
30293                                   ;       |  DB     MONTH     ;(1-12)      Note: Range check is not performed.
30294                                   ;       |  DB     DATE      ;(1-31)            0 is filled when the corresponding field was not specified.
30295                                   ;       | or
30296                                   ;       |  DB     HOUR      ;(0-23)       IN CASE OF TIME
30297                                   ;       |  DB     MINUTES    ;(0-59)      Note: Range check is not performed .
30298                                   ;       |  DB     SECONDS    ;(0-59)           0 is filled when the corresponding field was not specified .
30299                                   ;       |  DB     HUNDREDTHS ;(0-99)
30300                                   ;       +-
30301                                   ;
30302                                   ;
30303                                   ;Note: ITEM_TAG is 0FFH when the caller does not specify the choice
30304                                   ;      list.
30305                                   ;
30306                                   ;      YEAR: If the input value for the year is less than 100, parser
30307                                   ;        adds 1900 to it.  For example, when 87 is input to parser for
30308                                   ;        the year value, he returns 1987.
30309                                   ; ---------------------------------------------------------------------------
30310
30311                                   ;-------------------------------- Result block structure
30312                                   struc _$P_Result_Blk
30313 00000000 ??                        .Type:        resb 1              ;AN000; Type returned
30314 00000001 ??                        .Item_Tag:    resb 1              ;AN000; Matched item tag
30315 00000002 ????                      .SYNONYM_Ptr: resw 1              ;AN000; pointer to Synonym list returned
30316 00000004 ????????                  .Picked_Val:  resb 4              ;AN000; value
30317                                   endstruc
30318
30319                                   ;--------------------------------
30320                                   ;**** values for the type field in the result block
30321
30322                                   _$P_EOL               equ 0             ;AN000; End of line
30323                                   _$P_Number     equ 1              ;AN000; Number
30324                                   _$P_List_Idx          equ 2             ;AN000; List Index
30325                                   _$P_String     equ 3              ;AN000; String
30326                                   _$P_Complex    equ 4              ;AN000; Complex
30327                                   _$P_File_Spec         equ 5             ;AN000; File Spec
30328                                   _$P_Drive      equ 6              ;AN000; Drive
30329                                   _$P_Date_F     equ 7              ;AN000; Date
30330                                   _$P_Time_F     equ 8              ;AN000; Time
30331                                   _$P_Quoted_String  equ    9              ;AN000; Quoted String
30332
30333                                   _$P_No_Tag     equ 0FFh           ;AN000; No ITEM_TAG found
30334
30335                                   ;**** Return code
30336                                   ;
30337                                   ; following return code will be returned in the AX register.
30338
30339                                   _$P_No_Error          equ 0             ;AN000; No error
30340                                   _$P_Too_Many          equ 1             ;AN000; Too many operands
30341                                   _$P_Op_Missing        equ 2             ;AN000; Required operand missing
30342                                   _$P_Not_In_SW         equ 3             ;AN000; Not in switch list provided
30343                                   _$P_Not_In_Key        equ 4             ;AN000; Not in keyword list provided
30344                                   _$P_Out_Of_Range   equ    6              ;AN000; Out of range specified
30345                                   _$P_Not_In_Val        equ 7             ;AN000; Not in value list provided
30346                                   _$P_Not_In_Str        equ 8             ;AN000; Not in string list provided
30347                                   _$P_Syntax     equ 9              ;AN000; Syntax error
30348                                   _$P_RC_EOL     equ -1             ;AN000; End of command line
30349
30350                                   ; DATA - Retro DOS v4.0 - 27/03/2019
30351
30352                                   ; MSDOS 6.2 IO.SYS SYSINIT:179Ch
30353
30354                                   ; 14/04/2024 - Retro DOS v5.0
30355                                   ; PCDOS 7.1 IBMBIO.COM - SYSINIT:1C62h
30356
30357                                   ;********************** Local Data ************************************
```

```
30358 0000196F 0000                   _$P_ORDINAL:          dw  0             ;AN000; Operand ordinal save area
30359 00001971 0000                   _$P_RC:               dw  0             ;AN000; Return code from parser
30360 00001973 0000                   _$P_SI_Save:          dw  0             ;AN000; Pointer of command buffer
30361 00001975 0000                   _$P_DX:               dw  0             ;AN000; Return result buffer address
30362 00001977 00                     _$P_Terminator:       db  0             ;AN000; Terminator code (ASCII)
30363 00001978 0000                   _$P_DBCSEV_OFF:       dw  0             ;AN000; Offset of DBCS EV
30364 0000197A 0000                   _$P_DBCSEV_SEG:       dw  0             ;AN000; Segment of DBCS EV
30365 0000197C 0000                   _$P_Flags:    dw  0             ;AN000; Parser internal flags
30366                                 %define _$P_Flags1 _$P_Flags            ;AN038; to reference first byte flags
30367                                 %define _$P_Flags2 _$P_Flags+1          ;AN038; to reference second byte flags only
30368
30369                                 ;in second byte of _$P_Flags, referenced as _$P_Flags2:
30370                                 _$P_equ               equ 01h          ;AN000; "=" packed in string buffet
30371                                 _$P_Neg               equ 02h          ;AN000; Negative value
30372                                 _$P_Time12    equ 04h          ;AN000; set when PM is specified
30373                                 _$P_Key_Cmp   equ 08h          ;AN000; set when keyword compare
30374                                 _$P_SW_Cmp    equ 10h          ;AN000; set when switch compare
30375                                 _$P_Extra     equ 20h          ;AN000; set when extra delimiter found
30376                                 _$P_SW                equ 40h          ;AN000; set when switch found (tm08)
30377                                 _$P_Signed    equ 80h          ;AN000; signed numeric specified
30378
30379                                 ;in first byte of _$P_Flags, referenced as _$P_Flags1:
30380                                 _$P_time12am          equ 01h          ;AN038; set when AM is specified on time
30381                                 _$P_TIME_AGAIN        equ 02h          ;AN039; SET WHEN READY TO RE-PARSE TIME
30382
30383 0000197E 0000                   _$P_SaveSI_Cmpx:   dw     0             ;AN000; save si for later use by complex
30384 00001980 0000                   _$P_KEYorSW_Ptr:   dw     0             ;AN000; points next to "=" or ":" code
30385 00001982 0000                   _$P_Save_EOB:         dw  0             ;AN000; save pointer to EOB
30386 00001984 0000                   _$P_Found_SYNONYM: dw     0             ;AN000; es:@ points to found synonym
30387
30388 00001986 00<rep 80h>            _$P_STRING_BUF:       times 128 db 0    ;AN000; Pick a operand from command line
30389                                 _$P_STRING_BUF_END equ    $             ;AN000;
30390
30391                                 ; 25/10/2022
30392                                 ; (MSDOS 5.0 IO.SYS, SYSINIT:16F8h)
30393
30394 00001A06 FF                     _$P_Char_CAP_Ptr:  db     0FFh          ;AN000; info id
30395 00001A07 0000                               dw  0             ;AN000; offset of char case map table
30396 00001A09 0000                               dw  0             ;AN000; segment of char case map table
30397                                 ; 25/10/2022
30398                                 ;IF CAPSW
30399                                 ;_$P_File_CAP_Ptr: db     0FFh          ;AN000; info id
30400                                 ;             dw  0             ;AN000; offset of file case map table
30401                                 ;             dw  0             ;AN000; segment of file case map table
30402                                 ;ENDIF
30403
30404                                 ; (tm06) IF FileSW                ;AN000;(Check if file spec is supported)
30405                                 ;
30406
30407                                 ;M029
30408                                 ;!!!WARNING!!!
30409                                 ; In routine SYSPARSE (parse.asm), _$P_FileSp_Char is reinitialized using
30410                                 ;hardcoded strings. If the chars in the string are changed here, corresponding
30411                                 ;changes need to be made in SYSPARSE
30412
30413                                 ;IF FileSW+DrvSW                  ;AN000;(Check if file spec is supported)
30414
30415                                 ; 25/10/2022
30416                                 ; (MSDOS 5.0 IO.SYS, SYSINIT:16FDh)
30417
30418 00001A0B 5B5D7C3C3E2B3D3B22       _$P_FileSp_Char       db  '[]|<>+=;"'     ;AN000; delimitter of file spec
30419                                 _$P_FileSp_Len        equ $-_$P_FileSp_Char ;AN000;
30420
30421                                 ;ENDIF                                  ;AN000;(of FileSW)
30422
30423                                 ; delimiter parsing
30424                                 _$P_colon_period   equ    01h           ;AN032; check for colon & period
30425                                 _$P_period_only       equ 02h           ;AN032; check only for period
30426
30427                                 ;filespec error flag
30428 00001A14 00                     _$P_err_flag:         db  0             ;AN033; flag set if filespec parsing error
30429                                                                         ;AN033;  was detected.
30430                                 _$P_error_filespec equ    01h           ;AN033; mask to set flag
30431
30432
30433                                 ; PARSE.ASM - MSDOS 6.0 - 1991
30434                                 ; ============================================================================
30435                                 ; 27/03/2019 - Retro DOS v4.0
30436
30437                                 ;****************************************************************************
30438                                 ; SysParse;
30439                                 ;
30440                                 ;  Function : Parser Entry
30441                                 ;
30442                                 ;  Input: DS:SI -> command line
30443                                 ;     ES:DI -> parameter block
30444                                 ;     cs -> psdata.inc
30445                                 ;     CX = operand ordinal
30446                                 ;
30447                                 ;     Note:  ES is the segment containing all the control blocks defined
30448                                 ;            by the caller, except for the DOS COMMAND line parms, which
30449                                 ;            is in DS.
30450                                 ;
30451                                 ;  Output: CY = 1   error of caller, means invalid parameter block or
30452                                 ;               invalid value list. But this parser does NOT implement
30453                                 ;               this feature. Therefore CY always zero.
30454                                 ;
30455                                 ;      CY = 0   AX = return code
30456                                 ;               BL = terminated delimiter code
30457                                 ;               CX = new operand ordinal
30458                                 ;               SI = set past scaned operand
30459                                 ;               DX = selected result buffer
30460                                 ;
30461                                 ; Use:      _$P_Skip_Delim, _$P_Chk_EOL, _$P_Chk_Delim, _$P_Chk_DBCS
30462                                 ;   _$P_Chk_Swtch, _$P_Chk_Pos_Control, _$P_Chk_Key_Control
30463                                 ;   _$P_Chk_Sw_Control, _$P_Fill_Result
30464                                 ;
30465                                 ; Vars: _$P_Ordinal(RW), _$P_RC(RW), _$P_SI_Save(RW), _$P_DX(R), _$P_Terminator(R)
30466                                 ;   _$P_SaveSI_Cmpx(W), _$P_Flags(RW), _$P_Found_SYNONYM(R), _$P_Save_EOB(W)
30467                                 ;
30468                                 ;-------- Modification History ----------------------------------------
30469                                 ;
30470                                 ;  4/04/87 : Created by K. K,
30471                                 ;  4/28/87 : _$P_Val_YH assemble error (tm01)
30472                                 ;      : JMP SHORT assemble error (tm02)
30473                                 ;  5/14/87 : Someone doesn't want to include psdata (tm03)
30474                                 ;  6/12/87 : _$P_Bridge is missing when TimeSw equ 0 and (CmpxSw equ 1 or
30475                                 ;         DateSW equ 1)          (tm04)
30476                                 ;  6/12/87 : _$P_SorD_Quote is missing when QusSw equ 0 and CmpxSW equ 1
30477                                 ;                                (tm05) in PSDATA.INC
30478                                 ;  6/12/87 : _$P_FileSp_Char and _$P_FileSP_Len are missing
30479                                 ;         when FileSW equ 0 and DrvSW equ 1 (tm06) in PSDATA.INC
30480                                 ;  6/18/87 : $VAL1 and $VAL3, $VAL2 and $VAL3 can be used in the same
30481                                 ;         value-list block       (tm07)
```

```
30482                           ;  6/20/87 : Add _$P_SW to check if there's an omiting parameter after
30483                           ;       switch (keyword) or not. If there is, backup si for next call
30484                           ;       (tm08)
30485                           ;  6/24/87 : Complex Item checking does not work correctly when CmpSW equ 1
30486                           ;       and DateSW equ 0 and TimeSW equ 0 (tm09)
30487                           ;  6/24/87 : New function flag _$P_colon_is_not_necessary for switch
30488                           ;       /+15 and /+:15 are allowed for user (tm10)
30489                           ;  6/29/87 : ECS call changes DS register but it causes the address problem
30490                           ;       in user's routines. _$P_Chk_DBCS (tm11)
30491                           ;  7/10/87 : Switch with no_match flag (0x0000H) does not work correctly
30492                           ;                                       (tm12)
30493                           ;  7/10/87 : Invalid switch/keyword does not work correctly
30494                           ;                                       (tm13)
30495                           ;  7/10/87 : Drive_only breaks 3 bytes after the result buffer
30496                           ;                                       (tm14)
30497                           ;  7/12/87 : Too_Many_Operands sets DX=0 as the PARSE result
30498                           ;                                       (tm15)
30499                           ;  7/24/87 : Negative lower bound on numeric ranges cause trouble
30500
30501                           ;  7/24/87 : Quoted strings being returned with quotes.
30502
30503                           ;  7/28/87 : Kerry S (;AN018;)
30504                           ;       Non optional value on switch (match flags<>0 and <>1) not flagged
30505                           ;       as an error when missing.       Solution: return error 2.  Modules
30506                           ;       affected: _$P_Chk_SW_Control.
30507
30508                           ;  7/29/87 : Kerry S (;AN019;)
30509                           ;       Now allow the optional bit in match flags for switches.  This
30510                           ;       allows the switch to be encountered with a value or without a
30511                           ;       value and no error is returned.
30512                           ;
30513
30514                           ;  8/28/87 : Ed K, Kerry S (;AN020;)
30515                           ;  9/14/87   In PROC _$P_Get_DecNum, when checking for field separators
30516                           ;       within a date response, instead of checking just for the one
30517                           ;       character defined by the COUNTRY DEPENDENT INFO, check for
30518                           ;       all three chars, "-", "/", and ".". Change _$P_Chk_Switch to allow
30519                           ;       slashes in date strings when DateSw (assembler switch) is set.
30520
30521                           ;  9/1/87  : Kerry S (;AN021)
30522                           ;       In PROC _$P_String_Comp, when comparing the switch or keyword on
30523                           ;       the command line with the string in the control block the
30524                           ;       comparing was stopping at a colon (switch) or equal (keyword)
30525                           ;       on the command line and assuming a match.      This allowed a shorter
30526                           ;       string on the command line than in the synonym list in the control
30527                           ;       block.  I put in a test for a null in the control block so the
30528                           ;       string in the control block must be the same length as the string
30529                           ;       preceeding the colon or equal on the command line.
30530
30531                           ;  8/28/87 : Kerry S (;AN022;)
30532                           ;       All references to data in PSDATA.INC had CS overrides.  This caused
30533                           ;       problems for people who included it themselves in a segment other
30534                           ;       than CS.  Added switch to allow including PSDATA.INC in any
30535                           ;       segment.
30536
30537                           ;  9/16/87 : Ed K (;AN023;) PTM1040
30538                           ;       in _$P_set_cdi PROC, it assumes CS points to psdata. Change Push CS
30539                           ;       into PUSH cs.  In _$P_Get_DecNum PROC, fix AN020
30540                           ;       forced both TIME and DATE to use the delims, "-","/",".".
30541                           ;       Created FLag, in _$P_time_Format PROC, to request the delim in
30542                           ;       BL be used if TIME is being parsed.
30543
30544                           ;  9/24/87 : Ed K
30545                           ;       Removed the include to STRUC.INC.      Replaced the STRUC macro
30546                           ;       invocations with their normally expanded code; made comments
30547                           ;       out of the STRUC macro invocation statements to maintain readability.
30548
30549                           ;  9/24/87 : Ed K (;AN024;) PTM1222
30550                           ;       When no CONTROL for a keyword found, tried to fill in RESULT
30551                           ;       pointed to by non-existant CONTROL.
30552
30553                           ; 10/15/87 : Ed K (;AN025;) PTM1672
30554                           ;       A quoted text string can be framed only by double quote.  Remove
30555                           ;       support to frame quoted text string with single quote.
30556                           ;       (apostrophe) _$P_SorD_Quote is removed from PSDATA.INC.
30557                           ;       _$P_SQuote EQU also removed from PSDATA.INC.  Any references to
30558                           ;       single quote in PROC prologues are left as is for history reasons.
30559
30560                           ;       This fixes another bug, not mentioned in p1672, in that two
30561                           ;       quote chars within a quoted string is supposed to be reported as
30562                           ;       one quote character, but is reported as two quotes.  This changed
30563                           ;       two instructions in PROC _$P_Quoted_Str.
30564
30565                           ;       Also fixed are several JMP that caused a NOP, these changed to
30566                           ;       have the SHORT operator to avoid the unneeded NOP.
30567
30568                           ;       The code and PSDATA.INC have been aligned for ease of reading.
30569
30570                           ; 10/26/87 : Ed K (;AN026;) PTM2041, DATE within SWITCH, BX reference to
30571                           ;       psdata buffer should have cs.
30572
30573                           ; 10/27/87 : Ed K (;AN027;) PTM2042 comma between keywords implies
30574                           ;       positional missing.
30575
30576                           ; 11/06/87 : Ed K (;AN028;) PTM 2315 Parser should not use line feed
30577                           ;       as a line delimiter, should use carriage return.
30578                           ;       Define switch: LFEOLSW, if on, accept LF as end of line char.
30579
30580                           ; 11/11/87 : Ed K (;AN029;) PTM 1651 GET RID OF WHITESPACE AROUND "=".
30581
30582                           ; 11/18/87 : Ed K (;AN030;) PTM 2551 If filename is just "", then
30583                           ;       endless loop since SI is returned still pointing to start
30584                           ;       of that parm.
30585
30586                           ; 11/19/87 : Ed K (;AN031;) PTM 2585 date & time getting bad values.
30587                           ;       Vector to returned string has CS instead of cs, but
30588                           ;       when tried to fix it on previous version, changed similar
30589                           ;       but wrong place.
30590
30591                           ; 12/09/87 : Bill L (;AN032;) PTM 2772 colon and period are now valid
30592                           ;       delimiters between hours, minutes, seconds for time. And period
30593                           ;       and comma are valid delimiters between seconds and 100th second.
30594
30595                           ; 12/14/87 : Bill L (;AN033;) PTM 2722 if illegal delimiter characters
30596                           ;       in a filespec, then flag an error.
30597
30598                           ; 12/22/87 : Bill L (;AN034;)         All local data to parser is now
30599                           ;       indexed off of the cs equate instead of the DS register.
30600                           ;       Using this method, DS can point to the segment of PSP or to psdata
30601                           ;  -->          local parser data. Why were some references to local data changed
30602                           ;       to do this before, but not all ?????
30603
30604                           ; 02/02/88 : Ed K (;AC035;) INSPECT utility, suggests optimizations.
30605
```

```
                                ; 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 _$P_Pack_End    ;AN000; 3/9 then
00001A87 7505                       jnz     short _$P_Pack_End_backup_si ;AN000; (tm08)

00001A89 E83506                     call    _$P_Skip_Delim        ;AN000; skip subsequent white space,too
00001A8C EB26                       jmp     short _$P_Pack_End    ;AN000; finish copy by placing NUL at end

                                _$P_Pack_End_backup_si:                   ;AN000; (tm08)
00001A8E 2EF606[7D19]41             test    byte [cs:_$P_Flags2],_$P_SW+_$P_equ ;AN000;AC034;  (tm08)
00001A94 741E                       jz      short _$P_Pack_End    ;AN000; (tm08)

00001A96 4E                         dec     si                    ;AN000; (tm08)
00001A97 EB1B                       jmp     short _$P_Pack_End    ;AN025; (tm08)

                                _$P_PL01:                         ;AN000;
00001A99 2E8807                     mov     [cs:bx],al            ;AN000; move byte to STRING_BUF
00001A9C 3C3D                       cmp     al,_$P_Keyword  ;'='  ;AN000; if it is equal character,
00001A9E 7506                       jne     short _$P_PL00                ;AN000; then

00001AA0 2E800E[7D19]01             or      byte [cs:_$P_Flags2],_$P_equ ;AC034; remember it in flag
                                _$P_PL00:                         ;AN000;
00001AA6 43                         inc     bx                    ;AN000; ready to see next byte
00001AA7 E8D506                     call    _$P_Chk_DBCS          ;AN000; was it 1st byte of DBCS ?
00001AAA 73C5                       jnc     short _$P_Pack_Loop   ;AN000; if no, process to next byte

00001AAC AC                         lodsb                         ;AN000; if yes, store
00001AAD 2E8807                     mov     [cs:bx],al            ;AN000;    2nd byte of DBCS
00001AB0 43                         inc     bx                    ;AN000; update pointer
00001AB1 EBBE                       jmp     short _$P_Pack_Loop   ;AN000; process to next byte

                                _$P_Pack_End_BY_EOL:                      ;AN000;
00001AB3 4E                         dec     si                    ;AN000; backup si pointer
                                _$P_Pack_End:                             ;AN000;
00001AB4 2E8936[7319]               mov     [cs:_$P_SI_Save],si     ;AC034; save next pointer, SI
                                    ; 07/09/2023
                                    ;mov    byte [cs:bx],_$P_NULL ;AN000; put NULL at the end
00001AB9 30E4                       xor     ah,ah ; 0 ; *
00001ABB 2E8827                     mov     [cs:bx],ah ; _$P_NULL ;AN000; put NULL at the end
                                    ;
00001ABE 2E891E[8219]               mov     [cs:_$P_Save_EOB],bx    ;AC034; 3/17/87 keep the address for later use of complex
                                    ;mov    bx,[es:di+_$P_PARMS_Blk.PARMSX_Address] ;AN000; get PARMSX address
00001AC3 268B1D                     mov     bx,[es:di]
                                    ;;lea   si,[cs:_$P_STRING_BUF];AC034;
                                    ; 02/11/2022
                                    ;lea    si,[_$P_STRING_BUF]
                                    ; 07/09/2023
00001AC6 BE[8619]                   mov     si,_$P_STRING_BUF
00001AC9 2E803C2F                   cmp     byte [cs:si],_$P_Switch ;AN000; the operand begins w/ switch char ?
00001ACD 7440                       je      short _$P_SW_Manager  ;AN000; if yes, process as switch

00001ACF 2E803C22                   cmp     byte [cs:si],_$P_DQuote       ;M005;is it a string?
00001AD3 7408                       je      short _$P_Positional_Manager ;M005;if so, process as one!

00001AD5 2EF606[7D19]01             test    byte [cs:_$P_Flags2],_$P_equ ;AC034; the operand includes equal char ?
00001ADB 7552                       jnz     short _$P_Key_Manager ;AN000; if yes, process as keyword

                                _$P_Positional_Manager:                   ;AN000; else process as positional
00001ADD 268A4701                   mov     al,[es:bx+_$P_PARMSX_Blk.MaxP] ;AN000; get maxp
                                    ; 07/09/2023
                                    ;xor    ah,ah                 ;AN000; ax = maxp
00001AE1 2E3906[6F19]               cmp     [cs:_$P_ORDINAL],ax   ;AC034; too many positional ?
00001AE6 7312                       jae     short _$P_Too_Many_Error ;AN000; if yes, set exit code to too many

00001AE8 2EA1[6F19]                 mov     ax,[cs:_$P_ORDINAL]   ;AC034; see what the current ordinal
00001AEC D1E0                       shl     ax,1                  ;AN000; ax = ax*2
00001AEE 43                         inc     bx                    ;AC035; add '2' to
00001AEF 43                         inc     bx                    ;AC035;  BX reg
                                                                  ;AN000; now bx points to 1st CONTROL
00001AF0 01C3                       add     bx,ax                 ;AN000; now bx points to specified CONTROL address
00001AF2 268B1F                     mov     bx,[es:bx]            ;AN000; now bx points to specified CONTROL itself
00001AF5 E87200                     call    _$P_Chk_Pos_Control   ;AN000; Do process for positional
00001AF8 EB53                       jmp     short _$P_Return_to_Caller ;AN000; and return to the caller

                                _$P_Too_Many_Error:                       ;AN000;
00001AFA 2EC706[7119]0100           mov     word [cs:_$P_RC],_$P_Too_Many ;AC034; set exit code
00001B01 EB4A                       jmp     short _$P_Return_to_Caller ;AN000; and return to the caller

                                    ; 07/09/2023 - Retro DOSD v4.2 IO.SYS (Optimization)
                                    ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:1E06h)
                                get_maxp:
                                    ;mov    al,[es:bx+1]
00001B03 268A4701                   mov     al,[es:bx+_$P_PARMSX_Blk.MaxP] ;AN000; get maxp
                                    ; 07/09/2023
                                    ; ah=0 ; *
                                    ;xor    ah,ah ; 0             ;AN000; ax = maxp
00001B07 30ED                       xor     ch,ch ; **
00001B09 40                         inc     ax                    ;AN000;
00001B0A D1E0                       shl     ax,1                  ;AN000; ax = (ax+1)*2
00001B0C 01C3                       add     bx,ax                 ;AN000; now bx points to maxs
00001B0E C3                         retn

                                _$P_SW_Manager:                           ;AN000;
                                    ; 07/09/2023
                                    ;mov    al,[es:bx+_$P_PARMSX_Blk.MaxP] ;AN000; get maxp
                                    ;xor    ah,ah                 ;AN000; ax = maxp
                                    ;inc    ax                    ;AN000;
                                    ;shl    ax,1                  ;AN000; ax = (ax+1)*2
                                    ;add    bx,ax                 ;AN000; now bx points to maxs
00001B0F E8F1FF                     call    get_maxp ; 07/09/2023

00001B12 268A0F                     mov     cl,[es:bx]            ;AN000;
                                    ; 07/09/2023
                                    ;xor    ch,ch ; ** (ch=0)     ;AN000; cx = maxs
                                    ;or     cx,cx                 ;AN000; at least one switch ?
                                    ;jz     short _$P_SW_Not_Found        ;AN000;
                                    ; 07/07/2023
00001B15 E30F                       jcxz    _$P_SW_Not_Found      ; no

00001B17 43                         inc     bx                    ;AN000; now bx points to 1st CONTROL address

                                _$P_SW_Mgr_Loop:                  ;AN000;
00001B18 53                         push    bx                    ;AN000;
00001B19 268B1F                     mov     bx,[es:bx]            ;AN000; bx points to Switch CONTROL itself
00001B1C E8A900                     call    _$P_Chk_SW_Control    ;AN000; do process for switch
00001B1F 5B                         pop     bx                    ;AN000;
00001B20 732B                       jnc     short _$P_Return_to_Caller ;AN000; if the CONTROL is for the switch, exit

00001B22 43                         inc     bx                    ;AC035; add '2' to
00001B23 43                         inc     bx                    ;AC035;  BX reg
                                                                  ;AN000; else bx points to the next CONTROL
00001B24 E2F2                       loop    _$P_SW_Mgr_Loop               ;AN000; and loop

                                _$P_SW_Not_Found:                 ;AN000;
00001B26 2EC706[7119]0300           mov     word [cs:_$P_RC],_$P_Not_In_SW ;AC034; here no CONTROL for the switch has
00001B2D EB1E                       jmp     short _$P_Return_to_Caller ;AN000; not been found, means error.
```

```
                                _$P_Key_Manager:                ;AN000;
                                    ; 07/09/2023
                                    ;mov    al,[es:bx+_$P_PARMSX_Blk.MaxP] ;AN000; get maxp
                                    ;xor    ah,ah                  ;AN000; ax = maxp
                                    ;inc    ax                     ;AN000;
                                    ;shl    ax,1                   ;AN000; ax = (ax+1)*2
                                    ;add    bx,ax                  ;AN000; now bx points to maxs
00001B2F E8D1FF                     call    get_maxp ; 07/09/2023

00001B32 268A07                     mov     al,[es:bx]             ;AN000;
00001B35 30E4                       xor     ah,ah ; 0              ;AN000; ax = maxs
00001B37 D1E0                       shl     ax,1                   ;AN000;
00001B39 40                         inc     ax                     ;AN000; ax = ax*2+1
00001B3A 01C3                       add     bx,ax                  ;AN000; now bx points to maxk
00001B3C 268A0F                     mov     cl,[es:bx]             ;AN000;
                                    ; 07/09/2023
                                    ;xor    ch,ch ; ** (ch=0)      ;AN000; cx = maxk
                                    ;or     cx,cx                  ;AN000; at least one keyword ?
                                    ;jz     short _$P_Key_Not_Found       ;AN000;
                                    ; 07/07/2023
00001B3F E305                       jcxz    _$P_Key_Not_Found      ; no

00001B41 43                         inc     bx                     ;AN000; now bx points to 1st CONTROL

                                _$P_Key_Mgr_Loop:               ;AN000;
                                    ; 07/09/2023
                                    ; ('_$P_Chk_Key_Control' contains only 'stc' instruction)
                                    ; (always returns with cf=1)
                                    ;push   bx                     ;AN000;
                                    ;mov    bx,[es:bx]             ;AN000; bx points to keyword CONTROL itself
                                    ;call   _$P_Chk_Key_Control    ;AN000; do process for keyword
                                    ;pop    bx                     ;AN000;
                                    ;jnc    short _$P_Return_to_Caller ;AN000; if the CONTROL is for the keyword, exit
                                    ; 07/09/2023
                                    ; cf=1 (after 'call _$P_Chk_Key_Control')

00001B42 43                         inc     bx                     ;AC035; add '2' to
00001B43 43                         inc     bx                     ;AC035;  BX reg
                                                                   ;AN000; else bx points to the next CONTROL
00001B44 E2FC                       loop    _$P_Key_Mgr_Loop       ;AN000; and loop

                                _$P_Key_Not_Found:              ;AN000;
00001B46 2EC706[7119]0400           mov     word [cs:_$P_RC],_$P_Not_In_Key ;AC034; here no CONTROL for the keyword has
                                _$P_Return_to_Caller:                ;AN000;
00001B4D 5D                         pop     bp                     ;AN000;
00001B4E 5F                         pop     di                     ;AN000;
00001B4F 5B                         pop     bx                     ;AN000;
00001B50 2E8B0E[6F19]               mov     cx,[cs:_$P_ORDINAL]    ;AC034; return next ordinal
00001B55 2EA1[7119]                 mov     ax,[cs:_$P_RC]         ;AC034; return exit code
00001B59 2E8B36[7319]               mov     si,[cs:_$P_SI_Save]    ;AC034; return next operand pointer
00001B5E 2E8B16[7519]               mov     dx,[cs:_$P_DX]         ;AC034; return result buffer address
00001B63 2E8A1E[7719]               mov     bl,[cs:_$P_Terminator];AC034; return delimiter code found
                                _$P_Single_Exit:                ;AN000;
00001B68 F8                         clc                            ;AN000;
00001B69 C3                         retn                           ;AN000;

                                ;******************************************************************************
                                ; _$P_Chk_Pos_Control
                                ;
                                ; Function: Parse CONTROL block for a positional
                                ;
                                ; Input:     ES:BX -> CONTROL block
                                ;         cs:SI -> _$P_STRING_BUF
                                ;
                                ; Output:    None
                                ;
                                ; Use:       _$P_Fill_Result, _$P_Check_Match_Flags
                                ;
                                ; Vars: _$P_Ordinal(W), _$P_RC(W)
                                ;******************************************************************************

                                _$P_Chk_Pos_Control:
00001B6A 50                         push    ax                     ;AN000;
                                    ;mov    ax,[es:bx+_$P_Control_Blk.Match_Flag] ;AN000;
00001B6B 268B07                     mov     ax,[es:bx]
                                    ; 12/12/2022
00001B6E A802                       test    al,_$P_Repeat
                                    ;test   ax,_$P_Repeat          ;AN000; repeat allowed ?
00001B70 7505                       jnz     short _$P_CPC00               ;AN000; then do not increment ORDINAL

00001B72 2EFF06[6F19]               inc     word [cs:_$P_ORDINAL] ;AC034; update the ordinal
                                _$P_CPC00:                      ;AN000;
00001B77 2E803C00                   cmp     byte [cs:si],_$P_NULL ;AN000; no data ?
00001B7B 7517                       jne     short _$P_CPC01               ;AN000;

                                    ; 12/12/2022
00001B7D A801                       test    al,_$P_Optional
                                    ;test   ax,_$P_Optional                ;AN000; yes, then is it optional ?
00001B7F 7509                       jnz     short _$P_CPC02               ;AN000;

00001B81 2EC706[7119]0200           mov     word [cs:_$P_RC],_$P_Op_Missing ;AC034; no, then error 3/17/87
00001B88 EB0D                       jmp     short _$P_CPC_Exit    ;AN000;

                                _$P_CPC02:                      ;AN000;
00001B8A 50                         push    ax                     ;AN000;
                                    ;mov    al,_$P_String          ;AN000; if it is optional return NULL
                                    ;mov    ah,_$P_No_Tag          ;AN000; no item tag indication
                                    ; 07/07/2023
00001B8B B803FF                     mov     ax,(_$P_No_Tag<<8)|_$P_String
00001B8E E89600                     call    _$P_Fill_Result               ;AN000;
00001B91 58                         pop     ax                     ;AN000;
00001B92 EB03                       jmp     short _$P_CPC_Exit    ;AN000;

                                _$P_CPC01:                      ;AN000;
00001B94 E81101                     call    _$P_Check_Match_Flags ;AN000;
                                _$P_CPC_Exit:                        ;AN000;
00001B97 58                         pop     ax                     ;AN000;
00001B98 C3                         retn                           ;AN000;

                                ;******************************************************************************
                                ; _$P_Chk_Key_Control
                                ;
                                ; Function: Parse CONTROL block for a keyword
                                ;
                                ; Input:     ES:BX -> CONTROL block
                                ;         cs:SI -> _$P_STRING_BUF
                                ;
                                ; Output:    CY = 1 : not match
                                ;
                                ; Use:       _$P_Fill_Result, _$P_Search_KEYorSW, _$P_Check_Match_Flags
                                ;
                                ; Vars: _$P_RC(W), _$P_SaveSI_Cmpx(W), _$P_KEYorSW_Ptr(R), _$P_Flags(W)
                                ;******************************************************************************
```

```
30977
30978                                   ; 07/09/2023
30979                                   ;_$P_Chk_Key_Control:
30980                                   ;   stc                         ;AN000; this logic works when the KeySW
30981                                   ;   retn                        ;AN000; is reset.
30982
30983                                   ;****************************************************************************
30984                                   ; _$P_Search_KEYorSW:
30985                                   ;
30986                                   ; Function: Seach specified keyword or switch from CONTROL
30987                                   ;
30988                                   ; Input:     ES:BX -> CONTROL block
30989                                   ;        cs:SI -> _$P_STRING_BUF
30990                                   ;
30991                                   ; Output:    CY = 1 : not match
30992                                   ;
30993                                   ; Use:       _$P_String_Comp, _$P_MoveBP_NUL, _$P_Found_SYNONYM
30994                                   ;****************************************************************************
30995
30996                                       ; 25/10/2022 - Retro DOS v4.0
30997                                       ; (MSDOS 5.0 IO.SYS - SYSINIT:18B6h)
30998
30999                                   _$P_Search_KEYorSW:                      ;AN000;
31000 00001B99 55                         push    bp                    ;AN000;
31001 00001B9A 51                         push    cx                    ;AN000;
31002 00001B9B 268A4F08                   mov     cl,[es:bx+_$P_Control_Blk.nid] ;AN000; Get synonym count
31003 00001B9F 30ED                       xor     ch,ch                 ;AN000; and set it to cx
31004                                     ;or     cx,cx                 ;AN000; No synonyms specified ?
31005                                     ;jz     short _$P_KEYorSW_Not_Found ;AN000; then indicate not found by CY
31006                                     ; 07/07/2023
31007 00001BA1 E30D                       jcxz    _$P_KEYorSW_Not_Found
31008
31009                                     ;lea    bp,[es:bx+_$P_Control_Blk.KEYorSW] ;AN000; BP points to the 1st synonym
31010                                     ; 25/10/2022
31011 00001BA3 8D6F09                     lea     bp,[bx+_$P_Control_Blk.KEYorSW]
31012                                     ;lea    bp,[bx+9]
31013                                   _$P_KEYorSW_Loop:                ;AN000;
31014 00001BA6 E8B503                     call    _$P_String_Comp             ;AN000; compare string in buffer w/ the synonym
31015 00001BA9 7308                       jnc     short _$P_KEYorSW_Found     ;AN000; If match, set it to synonym pointer
31016
31017 00001BAB E80E00                     call    _$P_MoveBP_NUL        ;AN000; else, bp points to the next string
31018 00001BAE E2F6                       loop    _$P_KEYorSW_Loop      ;AN000; loop nid times
31019                                   _$P_KEYorSW_Not_Found:                   ;AN000;
31020 00001BB0 F9                         stc                           ;AN000; indicate not found in synonym list
31021 00001BB1 EB06                       jmp     short _$P_KEYorSW_Exit;AN000; and exit
31022
31023                                   _$P_KEYorSW_Found:               ;AN000;
31024 00001BB3 2E892E[8419]               mov     [cs:_$P_Found_SYNONYM],bp ;AC034; set synonym pointer
31025 00001BB8 F8                         clc                           ;AN000; indicate found
31026                                   _$P_KEYorSW_Exit:                ;AN000;
31027 00001BB9 59                         pop     cx                    ;AN000;
31028 00001BBA 5D                         pop     bp                    ;AN000;
31029 00001BBB C3                         retn                          ;AN000;
31030
31031                                   ;****************************************************************************
31032                                   ; _$P_MoveBP_NUL
31033                                   ;****************************************************************************
31034
31035                                   _$P_MoveBP_NUL:
31036                                   _$P_MBP_Loop:                            ;AN000;
31037                                       ; 11/12/2022
31038 00001BBC 26807E0000                 cmp     byte [es:bp],_$P_NULL ;AN000; Increment BP that points
31039                                     ; 25/10/2022 (MSDOS 5.0 IO.SYS compatibility)
31040                                     ; (SYSINIT:18DBh)
31041                                     ;cmp     byte [es:bp+0],0
31042 00001BC1 7403                       je      short _$P_MBP_Exit    ;AN000; to the synomym list
31043
31044 00001BC3 45                         inc     bp                    ;AN000; until
31045 00001BC4 EBF6                       jmp     short _$P_MBP_Loop    ;AN000; NULL encountered.
31046
31047                                   _$P_MBP_Exit:                            ;AN000;
31048 00001BC6 45                         inc     bp                    ;AN000; bp points to next to NULL
31049 00001BC7 C3                         retn                          ;AN000;
31050
31051                                   ;****************************************************************************
31052                                   ; _$P_Chk_SW_Control
31053                                   ;
31054                                   ; Function: Parse CONTROL block for a switch
31055                                   ;
31056                                   ; Input:     ES:BX -> CONTROL block
31057                                   ;        cs:SI -> _$P_STRING_BUF
31058                                   ;
31059                                   ; Output:    CY = 1 : not match
31060                                   ;
31061                                   ; Use:       _$P_Fill_Result, _$P_Search_KEYorSW, _$P_Check_Match_Flags
31062                                   ;
31063                                   ; Vars:  _$P_SaveSI_Cmpx(W), _$P_KEYorSW_Ptr(R), _$P_Flags(W)
31064                                   ;****************************************************************************
31065
31066                                   _$P_Chk_SW_Control:
31067
31068                                   ;IF SwSW                                 ;AN000;(Check if switch is supported)
31069                                       ;or     byte [cs:_$P_Flags+1],10h
31070 00001BC8 2E800E[7D19]10             or      byte [cs:_$P_Flags2],_$P_SW_Cmp ;AC034; Indicate switch for later string comparison
31071 00001BCE E8C8FF                     call    _$P_Search_KEYorSW    ;AN000; Search the switch in the CONTROL block
31072 00001BD1 7248                       jc      short _$P_Chk_SW_Err0 ;AN000; not found, then try next CONTROL
31073
31074                                     ;and    [cs:_$P_Flags+],0EFh
31075 00001BD3 2E8026[7D19]EF             and     byte [cs:_$P_Flags2],0FFh-_$P_SW_Cmp
31076                                                                     ;AC034; reset the indicator previously set
31077 00001BD9 50                         push    ax                    ;AN000;             /switch:
31078 00001BDA 2EA1[8019]                 mov     ax,[cs:_$P_KEYorSW_Ptr] ;AC034;           ^       ^
31079 00001BDE 29F0                       sub     ax,si                 ;AN000;  SI    KEYorSW
31080 00001BE0 2E0106[7E19]               add     [cs:_$P_SaveSI_Cmpx],ax      ;AC034; update for complex list
31081 00001BE5 58                         pop     ax                    ;AN000;
31082
31083 00001BE6 2E8B36[8019]               mov     si,[cs:_$P_KEYorSW_Ptr] ;AC034; set si at the end or colon
31084 00001BEB 2E803C00                   cmp     byte [cs:si],_$P_NULL ;AN000; any data after colon
31085 00001BEF 7525                       jne     short _$P_CSW00              ;AN000; if yes, process match flags
31086
31087 00001BF1 2E807CFF3A                 cmp     byte [cs:si-1],_$P_Colon ;AN000; if no, the switch terminated by colon ?
31088 00001BF6 7509                       jne     short _$P_Chk_if_data_required ;AN000; if yes,
31089
31090 00001BF8 2EC706[7119]0900           mov     word [cs:_$P_RC],_$P_Syntax ;AC034; return syntax error
31091 00001BFF EB1C                       jmp     short _$P_Chk_SW_Exit ;AN000;
31092
31093                                   _$P_Chk_if_data_required:         ;AN018; no data, no colon
31094                                       ;cmp    word [es:bx+_$P_Control_Blk.Match_Flag],0
31095 00001C01 26833F00                   cmp     word [es:bx],0        ;AN018; should have data? zero match flag means switch followed by nothing is
OK
31096 00001C05 7416                       je      short _$P_Chk_SW_Exit ;AN018; match flags not zero so should have something if optional bit is not
on
31097
31098                                     ;test   word [es:bx+_$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;
```

```
31223 00001C84 26F6470201               test    byte [es:bx+_$P_Control_Blk.Function_Flag],_$P_CAP_File
31224                                                           ;AN000; need CAPS by file table?
31225 00001C89 7404                     jz      short _$P_RLT_CAP00   ;AN000;
31226
31227 00001C8B B004                     mov     al,_$P_DOSTBL_File    ;AN000; use file upper case table
31228 00001C8D EB09                     jmp     short _$P_RLT_CAP02   ;AN000;
31229
31230                                 _$P_RLT_CAP00:                          ;AN000;
31231 00001C8F 26F6470202               test    byte [es:bx+_$P_Control_Blk.Function_Flag],_$P_CAP_Char
31232                                                           ;AN000; need CAPS by char table ?
31233 00001C94 7405                     jz      short _$P_RLT_CAP01   ;AN000;
31234
31235 00001C96 B002                     mov     al,_$P_DOSTBL_Char    ;AN000; use character upper case table
31236                                 _$P_RLT_CAP02:                          ;AN000;
31237 00001C98 E8DF00                   call    _$P_Do_CAPS_String    ;AN000;  process CAPS along the table
31238                                 _$P_RLT_CAP01:                          ;AN000;
31239 00001C9B 58                       pop     ax                    ;AN000;
31240 00001C9C 26F6470210               test    byte [es:bx+_$P_Control_Blk.Function_Flag],_$P_Rm_Colon
31241                                                           ;AN000; removing colon at end ?
31242 00001CA1 7403                     jz      short _$P_RLT_Exit    ;AN000;
31243
31244 00001CA3 E8AE00                   call    _$P_Remove_Colon      ;AN000; then process it.
31245                                 _$P_RLT_Exit:                           ;AN000;
31246 00001CA6 5F                       pop     di                    ;AN000;
31247 00001CA7 C3                       retn                          ;AN000;
31248
31249                                 ;****************************************************************************
31250                                 ; _$P_Check_Match_Flags
31251                                 ;
31252                                 ; Function:  Check the mutch_flags and make the exit code and set the
31253                                 ;        result buffer
31254                                 ;
31255                                 ;       Check for types in this order:
31256                                 ;          Complex
31257                                 ;          Date
31258                                 ;          Time
31259                                 ;          Drive
31260                                 ;          Filespec
31261                                 ;          Quoted String
31262                                 ;          Simple String
31263                                 ;
31264                                 ; Input:     cs:SI -> _$P_STRING_BUF
31265                                 ;        ES:BX -> CONTROL block
31266                                 ;
31267                                 ; Output:    None
31268                                 ;
31269                                 ; Use:           _$P_Value, P$_SValue, _$P_Simple_String, _$P_Date_Format
31270                                 ;        _$P_Time_Format, _$P_Complex_Format, _$P_File_Foemat
31271                                 ;        _$P_Drive_Format
31272                                 ;****************************************************************************
31273
31274                                     ; 25/10/2022 - Retro DOS v4.0
31275                                     ; (MSDOS 5.0 IO.SYS - SYSINIT:19CFh)
31276
31277                                     ; 14/04/2024 - Retro DOS v5.0
31278                                     ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:1FC3h)
31279
31280                                     ; 12/12/2022
31281                                 _$P_Check_Match_Flags:
31282 00001CA8 2EC606[141A]00           mov     byte [cs:_$P_err_flag],_$P_NULL
31283                                                           ;AN033;AC034;; clear filespec error flag.
31284 00001CAE 50                       push    ax                    ;AN000;
31285                                   ;mov    ax,[es:bx+_$P_Control_Blk.Match_Flag]
31286 00001CAF 268B07                   mov     ax,[es:bx]            ;AN000; load match flag(16bit) to ax
31287 00001CB2 09C0                     or      ax,ax                 ;AC035; test ax for zero
31288 00001CB4 7517                     jnz     short _$P_Mat         ;AN000; (tm12)
31289 00001CB6 50                       push    ax                    ;AN000; (tm12)
31290 00001CB7 53                       push    bx                    ;AN000; (tm12)
31291 00001CB8 52                       push    dx                    ;AN000; (tm12)
31292 00001CB9 57                       push    di                    ;AN000; (tm12)
31293 00001CBA 2EC706[7119]0900         mov     word [cs:_$P_RC],_$P_Syntax ;AC034; (tm12)
31294                                   ;mov    ah,_$P_No_Tag         ;AN000; (tm12)
31295                                   ;mov    al,_$P_String         ;AN000; (tm12)
31296                                   ; 07/07/2023
31297 00001CC1 B803FF                   mov     ax,(_$P_No_Tag<<8)|_$P_String
31298 00001CC4 E860FF                   call    _$P_Fill_Result               ;AN000; (tm12)
31299 00001CC7 5F                       pop     di                    ;AN000; (tm12)
31300 00001CC8 5A                       pop     dx                    ;AN000; (tm12)
31301 00001CC9 5B                       pop     bx                    ;AN000; (tm12)
31302 00001CCA 58                       pop     ax                    ;AN000; (tm12)
31303                                   ; 12/12/2022
31304                                   ;jmp    short _$P_Bridge      ;AC035; (tm12)
31305                                   ; 12/12/2022
31306                                 ;_$P_Mat:                       ;AN000; (tm12)
31307                                   ;jmp    short _$P_Match03     ;AN025; (tm09)
31308                                 _$P_Bridge:
31309 00001CCB EB6E                     jmp     short _$P_Match_Exit  ;AN000; (tm02)
31310
31311                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
31312                                   ; (SYSINIT:19F9h)
31313                                   ; 12/12/2022
31314                                   ;nop    ; db 90h
31315
31316                                 ; 12/12/2022
31317                                 _$P_Mat:
31318                                 _$P_Match03:                            ;AN000;
31319                                   ;test   ax,_$P_Num_Val ; 8000h;AN000; Numeric value
31320                                   ; 07/07/2023
31321 00001CCD F6C480                   test    ah,(_$P_Num_Val>>8) ; 80h
31322 00001CD0 7412                     jz      short _$P_Match04     ;AN000;
31323
31324 00001CD2 2EC706[7119]0000         mov     word [cs:_$P_RC],_$P_No_Error ;AC034; assume no error
31325 00001CD9 E81E01                   call    _$P_Value             ;AN000; do process
31326 00001CDC 2E833E[7119]09           cmp     word [cs:_$P_RC],_$P_Syntax ;AC034; if error, examine the next type
31327 00001CE2 7557                     jne     short _$P_Match_Exit  ;AN000;
31328                                 _$P_Match04:                            ;AN000;
31329                                   ;test   ax,_$P_SNum_Val ; 4000h       ;AN000; Signed numeric value
31330                                   ; 07/07/2023
31331 00001CE4 F6C440                   test    ah,(_$P_SNum_Val>>8) ; 40h
31332 00001CE7 7412                     jz      short _$P_Match05     ;AN000;
31333
31334 00001CE9 2EC706[7119]0000         mov     word [cs:_$P_RC],_$P_No_Error ;AC034; assume no error
31335 00001CF0 E8E300                   call    _$P_SValue            ;AN000; do process
31336 00001CF3 2E833E[7119]09           cmp     word [cs:_$P_RC],_$P_Syntax ;AC034; if error, examine the next type
31337 00001CF9 7540                     jne     short _$P_Match_Exit  ;AN000;
31338                                 _$P_Match05:                            ;AN000;
31339                                   ;test   ax,_$P_Drv_Only ; 100h;AN000; Drive only
31340                                   ; 07/07/2023
31341 00001CFB F6C401                   test    ah,(_$P_Drv_Only>>8) ; 1
31342 00001CFE 7415                     jz      short _$P_Match06     ;AN000;
31343
31344 00001D00 2EC706[7119]0000         mov     word [cs:_$P_RC],_$P_No_Error ;AC034; assume no error
31345 00001D07 E8F202                   call    _$P_File_Format               ;AN000; 1st, call file format
31346 00001D0A E87203                   call    _$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
                        _$P_Sval01:                             ;AN000;
00001DF5 E80200                 call    _$P_Value               ;AN000; and process value
00001DF8 58                     pop     ax                      ;AN000;
00001DF9 C3                     retn

                        ;*****************************************************************************

                                ; 07/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                ; (MSDOS 6.21 IO.SYS - SYSINIT:1C6Ah)

                                ; 26/10/2022
                        _$P_Value:                              ;AN000;
00001DFA 50                     push    ax                      ;AN000;
00001DFB 51                     push    cx                      ;AN000;
00001DFC 52                     push    dx                      ;AN000;
00001DFD 56                     push    si                      ;AN000;
00001DFE 31C9                   xor     cx,cx                   ;AN000; cx = higher 16 bits
00001E00 31D2                   xor     dx,dx                   ;AN000; dx = lower 16 bits
00001E02 53                     push    bx                      ;AN000; save control pointer
                        _$P_Value_Loop:                                 ;AN000;
00001E03 2E8A04                 mov     al,[cs:si]              ;AN000; get character
00001E06 08C0                   or      al,al                   ;AN000; end of line ?
00001E08 7438                   jz      short _$P_Value00       ;AN000;

00001E0A E8DC00                 call    _$P_0099                ;AN000; make asc(0..9) to bin(0..9)
00001E0D 722F                   jc      short _$P_Value_Err0    ;AN000;

00001E0F 30E4                   xor     ah,ah                   ;AN000;
00001E11 89C5                   mov     bp,ax                   ;AN000; save binary number

                        ; 08/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
                        ; Ref: Disassembled PCDOS 7.1 IBMBIO.COM SYSINIT code
                        ;                               Erdogan Tan - July 2023
                        %if 0
00001E13 D1E5                   shl     dx,1                    ;AN000; to have 2*x
00001E15 D1D1                   rcl     cx,1                    ;AN000; shift left w/ carry
00001E17 E8E600                 call    _$P_Check_OVF           ;AN000; Overflow occurred ?
00001E1A 722A                   jc      short _$P_Value_Err0    ;AN000; then error, exit

00001E1C 89D3                   mov     bx,dx                   ;AN000; save low(2*x)
00001E1E 89C8                   mov     ax,cx                   ;AN000; save high(2*x)
00001E20 D1E2                   shl     dx,1                    ;AN000; to have 4*x
00001E22 D1D1                   rcl     cx,1                    ;AN000; shift left w/ carry
00001E24 E8D900                 call    _$P_Check_OVF           ;AN000; Overflow occurred ?
00001E27 721D                   jc      short _$P_Value_Err0    ;AN000; then error, exit

00001E29 D1E2                   shl     dx,1                    ;AN000; to have 8*x
00001E2B D1D1                   rcl     cx,1                    ;AN000; shift left w/ carry
00001E2D E8D000                 call    _$P_Check_OVF           ;AN000; Overflow occurred ?
00001E30 7214                   jc      short _$P_Value_Err0    ;AN000; then error, exit

00001E32 01D8                   add     dx,bx                   ;AN000; now have 10*x
00001E34 11C1                   adc     cx,ax                   ;AN000; 32bit ADD
00001E36 E8C700                 call    _$P_Check_OVF           ;AN000; Overflow occurred ?
00001E39 720B                   jc      short _$P_Value_Err0    ;AN000; then error, exit

00001E3B 01D2                   add     dx,bp                   ;AN000; Add the current one degree decimal
00001E3D 83D100                 adc     cx,0                    ;AN000; if carry, add 1 to high 16bit
00001E40 E8BD00                 call    _$P_Check_OVF           ;AN000; Overflow occurred ?
00001E43 7202                   jc      short _$P_Value_Err0    ;AN000; then error, exit

00001E45 46                     inc     si                      ;AN000; update pointer
00001E46 EBBB                   jmp     short _$P_Value_Loop    ;AN000; loop until NULL encountered
                        ;_$P_Value_Err0:
                        %endif
                        ;****
                        %if 1
                        ; 08/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
                                ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:2130h)

                                ; 14/04/2024 - Retro DOS v5.0
                                ;xor    ah,ah
                                ;mov    bp,ax                   ; save binary number

00001E48 E81C00                 call    _$P_Value_2x_OVF        ; multiply cx:dx by 2 and then check overflow
00001E4B 89D3                   mov     bx,dx                   ; ax:bx = 2*(cx:dx)
00001E4D 89C8                   mov     ax,cx
00001E4F E81500                 call    _$P_Value_2x_OVF        ; multiply cx:dx by 2 and then check overflow
00001E52 E81200                 call    _$P_Value_2x_OVF        ; multiply cx:dx by 2 and then check overflow
00001E55 01DA                   add     dx,bx                   ; 8*(cx:dx)+2*(cx:dx) = 10*(cx:dx)
00001E57 11C1                   adc     cx,ax
00001E59 E80F00                 call    _$P_Value_Chk_Add_OVF
00001E5C 01EA                   add     dx,bp                   ; Add the current one degree decimal
                                                                ; if carry, add 1 to high 16bit
00001E5E 83D100                 adc     cx,0
00001E61 E80700                 call    _$P_Value_Chk_Add_OVF   ; Overflow occurred ?
                                                                ; then error, exit (without return here)
00001E64 46                     inc     si                      ; update pointer
00001E65 EBD1                   jmp     short _$P_Value_Loop

                        _$P_Value_2x_OVF:
00001E67 D1E2                   shl     dx,1                    ; to have 2*x
00001E69 D1D1                   rcl     cx,1                    ; shift left w/ carry
                        _$P_Value_Chk_Add_OVF:
00001E6B E89E00                 call    _$P_Check_OVF           ; check overflow (for the last shift or add)
00001E6E 7201                   jc      short _$P_Value_OVF
00001E70 C3                     retn
                        _$P_Value_OVF:
00001E71 44                     inc     sp                      ; skip "call" return address to the caller
00001E72 44                     inc     sp

                        ;_$P_Value_Err0:
                        %endif
                        ;****

                        _$P_Value_Err0:                                 ;AN000;
00001E73 5B                     pop     bx                      ;AN000;
00001E74 E98300                 jmp     _$P_Value_Err           ;AN000; Bridge
                        ;
                        _$P_Value00:                                    ;AN000;
00001E77 5B                     pop     bx                      ;AN000; restore control pointer
00001E78 2EF606[7D19]02         test    byte [cs:_$P_Flags2],_$P_Neg ;AC034; here cx,dx = 32bit value
00001E7E 740A                   jz      short _$P_Value01       ;AN000; was it negative ?

00001E80 F7D1                   not     cx                      ;AN000; +
00001E82 F7D2                   not     dx                      ;AN000; |- Make 2's complement
00001E84 83C201                 add     dx,1                    ;AN000; |
00001E87 83D100                 adc     cx,0                    ;AN000; +

                        _$P_Value01:                                    ;AN000; / nval = 0
00001E8A 268B7706               mov     si,[es:bx+_$P_Control_Blk.Value_List] ;AN000; si points to value list
00001E8E 268A04                 mov     al,[es:si]              ;AN000; get nval
                                ; 07/09/2023
```

```
                                    ;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 ?
```

```
                                    jnz     short _$P_Sim00             ;AN000; then
                                
                                    mov     ah,_$P_No_Tag           ;AN000; No ITEM_TAG set
                                    jmp     short _$P_Sim_Exit      ;AN000; and set result buffer
                                
                                _$P_Sim00:                          ;AN000;
                                ;IF Val3SW+KeySW                    ;AN000;(Check if keyword or value list id #3 is supported)
                                    cmp     al,_$P_nval_String      ;AN000; String choice list provided ?
                                    jne     short _$P_Sim01             ;AN000; if no, syntax error
                                
                                    inc     di                      ;AN000;
                                    mov     al,[es:di]              ;AN000; al = nrng
                                    mov     ah,_$P_Len_Range        ;AN000;
                                    mul     ah                      ;AN000; Skip nrng field
                                    inc     ax                      ;AN000; ax = (nrng*9)+1
                                    add     di,ax                   ;AN000; di points to nnval
                                    mov     al,[es:di]              ;AN000; get nnval
                                    mov     ah,_$P_Len_Value        ;AN000;
                                    mul     ah                      ;AN000; Skip nnval field
                                    inc     ax                      ;AN000; ax = (nnval*5)+1
                                    add     di,ax                   ;AN000; di points to nstrval
                                    mov     al,[es:di]              ;AN000; get nstrval c
                                    inc     di                      ;AC035; add '2' to
                                    inc     di                      ;AC035;  DI reg
                                                                    ;AN000; di points to 1st string in list
                                _$P_Sim_Loop:                           ;AN000;
                                    mov     bp,[es:di]              ;AN000; get string pointer
                                    call    _$P_String_Comp             ;AN000; compare it with operand
                                    jnc     short _$P_Sim_Found     ;AN000; found on list ?
                                
                                    add     di,_$P_Len_String ; 3   ;AN000; if no, point to next choice
                                    dec     al                      ;AN000; loop nstval times in AL
                                    jne     short _$P_Sim_Loop      ;AN000;
                                                                    ;AN000; / Not found
                                    mov     word [cs:_$P_RC],_$P_Not_In_Str ;AC034;
                                    mov     ah,_$P_No_Tag           ;AN000; No ITEM_TAG set
                                    jmp     short _$P_Sim_Exit      ;AN000;
                                
                                _$P_Sim_Found:                          ;AN000;
                                    mov     ah,[es:di-1]            ;AN000; set item_tag
                                    mov     al,_$P_List_Idx             ;AN000;
                                    mov     dx,[es:di]              ;AN000; get address of STRING
                                    jmp     short _$P_Sim_Exit0     ;AN000;
                                ;ENDIF                                  ;AN000;(of Val3SW+KeySW)
                                _$P_Sim01:                          ;AN000;
                                    mov     word [cs:_$P_RC],_$P_Syntax ;AC034;
                                    mov     ah,_$P_No_Tag           ;AN000; No ITEM_TAG set
                                _$P_Sim_Exit:                           ;AN000;
                                    mov     al,_$P_String           ;AN000; Set type
                                _$P_Sim_Exit0:                          ;AN000;
                                    call    _$P_Fill_Result             ;AN000;
                                    pop     di                      ;AN000;
                                    pop     dx                      ;AN000;
                                    pop     bx                      ;AN000;
                                    pop     ax                      ;AN000;
                                    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
                                ;******************************************************************************
                                
                                _$P_String_Comp:
                                    push    ax                      ;AN000;
                                    push    bp                      ;AN000;
                                    push    dx                      ;AN000;
                                    push    si                      ;AN000;
                                    mov     dl,_$P_DOSTBL_Char      ;AN000; use character case map table
                                _$P_SCOM_Loop:                          ;AN000;
                                    mov     al,[cs:si]              ;AN000; get command character
                                    call    _$P_Chk_DBCS            ;AN000; DBCS ?
                                    jc      short _$P_SCOM00        ;AN000; yes,DBCS
                                
                                    call    _$P_Do_CAPS_Char        ;AN000; else, upper case map before comparison
                                ;IF KeySW+SwSW                          ;AN000;(Check if keyword or switch is supported)
                                    test    byte [cs:_$P_Flags2],_$P_Key_Cmp ;AC034; keyword search ?
                                    jz      short _$P_SCOM04        ;AN000;
                                
                                    cmp     al,_$P_Keyword          ;AN000; "=" is delimiter
                                    jne     short _$P_SCOM03        ;AN000; IF "=" on command line AND  (bp+1=> char after the "=" in synonym
                                list)
                                
                                    cmp     byte [es:bp+1],_$P_NULL ;AN021;  at end of keyword string in the control block THEN
                                    jne     short _$P_SCOM_Differ ;AN021;
                                
                                    jmp     short _$P_SCOM05        ;AN000; keyword found in synonym list
                                
                                _$P_SCOM04:                             ;AN000;
                                    test    byte [cs:_$P_Flags2],_$P_SW_Cmp ;AC034; switch search ?
                                    jz      short _$P_SCOM03        ;AN000;
                                
                                    cmp     al,_$P_Colon            ;AN000; ":" is delimiter, at end of switch on command line
                                    jne     short _$P_SCOM03        ;AN000; continue compares
                                
                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;cmp    byte [es:bp+0],_$P_NULL
                                    ; 11/12/2022
                                    cmp     byte [es:bp],_$P_NULL ;AN021; IF at end of switch on command AND
                                    jne     short _$P_SCOM_Differ ;AN021;   at end of switch string in the control block THEN
                                
                                _$P_SCOM05:                             ;AN000;   found a match
                                    inc     si                      ;AN000; si points to just after "=" or ":"
                                    jmp     short _$P_SCOM_Same     ;AN000; exit
                                
                                _$P_SCOM03:                             ;AN000;
                                ;ENDIF                                      ;AN000;(of KeySW+SwSW)
                                    ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;cmp    al,[es:bp+0]
                                    ; 11/12/2022
                                    cmp     al,[es:bp]              ;AN000; compare operand w/ a synonym
                                    jne     short _$P_SCOM_Differ0      ;AN000; if different, check ignore colon option
                                
                                    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;
00001FFF 2E8B3E[7E19]                   mov     di,[cs:_$P_SaveSI_Cmpx]       ;AC034; get user buffer address
                                _$P_FileF_Loop0:                        ;AN000; / skip special characters
00002004 2E8A04                         mov     al,[cs:si]              ;AN000; load character
00002007 08C0                           or      al,al                   ;AN000; end of line ?
00002009 7413                           jz      short _$P_FileF_Err     ;AN000; if yes, error exit

0000200B E85D00                         call    _$P_FileSp_Chk          ;AN000; else, check if file special character
0000200E 7523                           jne     short _$P_FileF03       ;AN000; if yes,

00002010 2EC606[141A]01                 mov     byte [cs:_$P_err_flag],_$P_error_filespec
                                                                        ;AN033;AC034;; set error flag- bad char.
00002016 5E                             pop     si                      ;AN033;
00002017 2EC60400                       mov     byte [cs:si],_$P_NULL   ;AN033;
0000201B 5F                             pop     di                      ;AN033;
0000201C EB3E                           jmp     short _$P_FileF02       ;AN033;

                                _$P_FileF_Err:                                  ;AN000;
0000201E 5E                             pop     si                      ;AN000;
0000201F 2EC60400                       mov     byte [cs:si],_$P_NULL   ;AN000;
00002023 5F                             pop     di                      ;AN000;

                                        ;test   word [es:bx+_$P_Control_Blk.Match_Flag],_$P_Optional ;AN000; is it optional ?
                                        ;test   byte [es:bx+_$P_Control_Blk.Match_Flag],_$P_Optional
                                        ; 12/12/2022
00002024 26F60701                       test    byte [es:bx],_$P_Optional
                                        ; 26/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                        ;test   word [es:bx],_$P_Optional
00002028 7532                           jnz     short _$P_FileF02       ;AN000;

0000202A 2EC706[7119]0200               mov     word [cs:_$P_RC],_$P_Op_Missing ;AC034; 3/17/87
00002031 EB29                           jmp     short _$P_FileF02       ;AN000;

                                _$P_FileF03:                                    ;AN000;
00002033 58                             pop     ax                      ;AN000; discard save si
00002034 56                             push    si                      ;AN000; save new si
                                _$P_FileF_Loop1:                        ;AN000;
00002035 2E8A04                         mov     al,[cs:si]              ;AN000; load character (not special char)
00002038 08C0                           or      al,al                   ;AN000; end of line ?
0000203A 741E                           jz      short _$P_FileF_RLT     ;AN000;

0000203C E82C00                         call    _$P_FileSp_Chk          ;AN000; File special character ?
0000203F 740B                           je      short _$P_FileF00       ;AN000;

00002041 E83B01                         call    _$P_Chk_DBCS            ;AN000; no, then DBCS ?
00002044 7302                           jnc     short _$P_FileF01       ;AN000;
00002046 47                             inc     di                      ;AN000; if yes, skip next byte
00002047 46                             inc     si                      ;AN000;
                                _$P_FileF01:                                    ;AN000;
00002048 47                             inc     di                      ;AN000;
00002049 46                             inc     si                      ;AN000;
0000204A EBE9                           jmp     short _$P_FileF_Loop1   ;AN000;
                                ;
                                _$P_FileF00:                                    ;AN000;
0000204C 2EA2[7719]                     mov     [cs:_$P_Terminator],al;AC034;
00002050 2EC60400                       mov     byte [cs:si],_$P_NULL   ;AN000; update end of string
00002054 47                             inc     di                      ;AN000;
00002055 2E893E[7319]                   mov     [cs:_$P_SI_Save],di     ;AC034; update next pointer in command line
                                _$P_FileF_RLT:                                  ;AN000;
0000205A 5E                             pop     si                      ;AN000;
0000205B 5F                             pop     di                      ;AN000;
                                _$P_FileF02:                                    ;AN000;
0000205C 58                             pop     ax                      ;AN000; (tm14)
                                        ;test   ax,_$P_File_Spc         ; 200h ;AN000; (tm14)
                                        ; 08/07/2023
0000205D F6C402                         test    ah,(_$P_File_Spc>>8) ; 2
00002060 7408                           jz      short _$P_Drv_Only_Exit       ;AN000; (tm14)

00002062 50                             push    ax                      ;AN000; (tm14)
                                        ;mov    ah,_$P_No_Tag           ;AN000; set
                                        ;mov    al,_$P_File_Spec        ;AN000; result
                                        ; 08/07/2023
00002063 B805FF                         mov     ax,(_$P_No_Tag<<8)|_$P_File_Spec ; 0FF05h
                                                                                ; set result
00002066 E8BEFB                         call    _$P_Fill_Result                 ;AN000; buffer to file spec
00002069 58                             pop     ax                      ;AN000;

                                _$P_Drv_Only_Exit:                      ;AN000; (tm14)
0000206A C3                             retn                            ;AN000;

                                ;******************************************************************************
                                ; _$P_FileSp_Chk
                                ;
                                ; Function:  Check if the input byte is one of file special characters
                                ;
                                ; Input:     cs:SI -> _$P_STRING_BUF
                                ;        AL = character code to be examineed
                                ;
                                ; Output:    ZF = 1 , AL is one of special characters
                                ;******************************************************************************

                                _$P_FileSp_Chk:
0000206B 53                             push    bx                      ;AN000;
0000206C 51                             push    cx                      ;AN000;
                                        ;;lea   bx,[cs:_$P_FileSp_Char] ;AC034; special character table
                                        ;lea    bx,[_$P_FileSp_Char]    ; "[]|<>+=;\"" at
                                                                        ; MSDOS 6.21 IO.SYS - SYSINIT:1838h
                                        ; 07/09/2023
0000206D BB[0B1A]                       mov     bx,_$P_FileSp_Char
00002070 B90900                         mov     cx,_$P_FileSp_Len ; 9 ;AN000; load length of it
                                _$P_FileSp_Loop:                        ;AN000;
00002073 2E3A07                         cmp     al,[cs:bx]              ;AN000; is it one of special character ?
00002076 7404                           je      short _$P_FileSp_Exit ;AN000;

00002078 43                             inc     bx                      ;AN000;
00002079 E2F8                           loop    _$P_FileSp_Loop                 ;AN000;

0000207B 41                             inc     cx                      ;AN000; reset ZF
                                _$P_FileSp_Exit:                        ;AN000;
0000207C 59                             pop     cx                      ;AN000;
0000207D 5B                             pop     bx                      ;AN000;
0000207E C3                             retn

                                ;ENDIF                                          ;AN000;(of FileSW+DrvSW)

                                ;IF DrvSW                               ;AN000;(Check if drive only is supported)

                                ;******************************************************************************
                                ; _$P_Drive_Format;
                                ;
                                ; Function:  Check if the input string is valid drive only format.
                                ;        And set the result buffer.
                                ;
                                ; Input:     cs:SI -> _$P_STRING_BUF
```

```
                                ;       ES:BX -> CONTROL block
                                ;
                                ; Output:    None
                                ;
                                ; Use:      _$P_Fill_Result, _$P_Chk_DBCS
                                ;
                                ; 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
```

```
                                _$P_Chk_Delim_Loop:                     ;AN000;
00002144 43                         inc     bx                      ;AN000;
00002145 263A01                     cmp     al,[es:bx+di]           ;AN000; Check extra Delim character
00002148 7406                       je      short _$P_Chk_Delim_Exit0 ;AN000;

0000214A E2F8                       loop    _$P_Chk_Delim_Loop      ;AN000; examine all extra delimiter

                                _$P_Chk_Delim_NZ:                       ;AN000;
                                    ;cmp    al,_$P_Space            ;AN000; reset ZF
                                    ; 08/07/2023
                                    ; cx=0 here
0000214C 41                         inc     cx ; cx=1, zf=0
                                _$P_Chk_Delim_Exit:                     ;AN000;
                                _$P_ChkDfin:                            ;AN000;
0000214D 59                         pop     cx                      ;AN000;
0000214E 5B                         pop     bx                      ;AN000;
0000214F C3                         retn                            ;AN000;

                                _$P_Chk_Delim_Exit0:                    ;AN000;
00002150 2EA2[7719]                 mov     [cs:_$P_Terminator],al ;AC034; keep terminated delimiter
00002154 2EF606[7D19]01             test    byte [cs:_$P_Flags2],_$P_equ ;AN027;AC034;; if terminating a key=
0000215A 7506                       jnz     short _$P_No_Set_Extra      ;AN027; then do not set the EXTRA bit

0000215C 2E800E[7D19]20             or      byte [cs:_$P_Flags2],_$P_Extra
                                                                        ;AC034; flag terminated extra delimiter or comma
                                _$P_No_Set_Extra:                       ;AN027;
00002162 38C0                       cmp     al,al                   ;AN000; set ZF
00002164 EBE7                       jmp     short _$P_Chk_Delim_Exit ;AN000;

                                ;******************************************************************************
                                ; _$P_Chk_Switch;
                                ;
                                ; Function: Check if AL is the switch character not in first position of
                                ;       _$P_STRING_BUF
                                ;
                                ; Input:    AL = character code
                                ;       BX = current pointer within _$P_String_Buf
                                ;       SI =>next char on command line (following the one in AL)
                                ;
                                ; Output:   CF = 1 (set)if AL is switch character, and not in first
                                ;           position, and has no chance of being part of a date string,
                                ;           i.e. should be treated as a delimiter.

                                ;       CF = 0 (reset, cleared) if AL is not a switch char, is in the first
                                ;           position, or is a slash but may be part of a date string, i.e.
                                ;           should not be treated as a delimiter.
                                ;
                                ; Vars:  _$P_Terminator(W)

                                ; Use:       _$P_0099
                                ;******************************************************************************

                                _$P_Chk_Switch:
                                    ;;lea   bp,[cs:_$P_STRING_BUF];AN020;AC034
                                    ;lea    bp,[_$P_STRING_BUF]    ;BP=OFFSET of _$P_String_Buf even in group addressing
                                    ; 08/07/2023
00002166 BD[8619]                   mov     bp,_$P_STRING_BUF

                                ;   .IF <BX NE BP> THEN             ;AN020;IF not first char THEN
00002169 39EB                       cmp     bx,bp                   ;AN000;
0000216B 7406                       je      short _$P_STRUC_L2      ;AN000;

                                ;   .IF <AL EQ _$P_Switch> THEN     ;AN020;otherwise see if a slash
0000216D 3C2F                       cmp     al,_$P_Switch           ;AN000;
0000216F 750C                       jne     short _$P_STRUC_L5      ;AN000;

00002171 F9                         stc                             ;AN020;not in first position and is slash
                                    ;jmp     short _$P_STRUC_L1     ;AN000;
                                    ; 12/12/2022
00002172 C3                         retn

                                ; 12/12/2022
                                ;_$P_STRUC_L5:                          ;AN000;
                                ;   CLC                             ;AN020;not a slash
                                ;;      .ENDIF                      ;AN020;
                                ;;  .ELSE                           ;AN020;is first char in the buffer, ZF=0
                                ;   jmp     short _$P_STRUC_L1      ;AN000;

                                _$P_STRUC_L2:                           ;AN000;
                                ;   .IF <AL EQ _$P_Switch> THEN     ;AN020;
00002173 3C2F                       cmp     al,_$P_Switch           ;AN000;
00002175 7506                       jne     short _$P_STRUC_L12     ;AN000;

00002177 2E800E[7D19]40             or      byte [cs:_$P_Flags2],_$P_SW ;AN020 ;AC034;;could be valid switch, first char and is slash
                                ;   .ENDIF                          ;AN020;

                                    ; 12/12/2022
                                    ; cf=0
                                    ;retn

                                _$P_STRUC_L5:
                                    ; 12/12/2022
                                _$P_STRUC_L12:                          ;AN000;
0000217D F8                         clc                             ;AN020;CF=0 indicating first char
                                ;   .ENDIF                          ;AN020;
                                _$P_STRUC_L1:                           ;AN000;
0000217E C3                         retn                            ;AN000;

                                ;******************************************************************************
                                ; _$P_Chk_DBCS:
                                ;
                                ;  Function: Check if a specified byte is in ranges of the DBCS lead bytes
                                ;
                                ;  Input:
                                ;     AL    = Code to be examineed
                                ;
                                ;  Output:
                                ;     If CF is on then a lead byte of DBCS
                                ;
                                ; Use: INT 21h w/AH=63
                                ;
                                ; Vars:  _$P_DBCSEV_Seg(RW), _$P_DBCSEV_Off(RW)
                                ;******************************************************************************

                                _$P_Chk_DBCS:
0000217F 1E                         push    ds                      ;AN000;
00002180 56                         push    si                      ;AN000;
00002181 53                         push    bx                      ;AN000; (tm11)
                                    ;cmp    word [cs:_$P_DBCSEV_SEG],0 ;AC034; ALREADY SET ?
                                    ;jne    short _$P_DBCS00        ;AN000;
                                    ; 08/07/2023
00002182 2E8B36[7A19]               mov     si,[cs:_$P_DBCSEV_SEG]
00002187 21F6                       and     si,si ; 0 ?
00002189 7525                       jnz     short _$P_DBCS00 ; already set
```

```
32710 0000218B 50                          push    ax                      ;AN000;
32711 0000218C 1E                          push    ds                      ;AN000; (tm11)
32712 0000218D 51                          push    cx                      ;AN000;
32713 0000218E 52                          push    dx                      ;AN000;
32714 0000218F 57                          push    di                      ;AN000;
32715 00002190 55                          push    bp                      ;AN000;
32716 00002191 06                          push    es                      ;AN000;
32717                                      ; si = 0 ; 08/07/2023
32718                                      ;xor    si,si                   ;AN000;
32719 00002192 8EDE                        mov     ds,si ; 0               ;AN000;
32720 00002194 B80063                      mov     ax,_$P_DOS_GetEV ; 6300h ;AN000; GET DBCS EV CALL
32721 00002197 CD21                        int     21h                     ;AN000;
32722                                              ; DOS - 3.2+ only - GET DOUBLE BYTE CHARACTER SET LEAD TABLE
32723 00002199 8CDB                        mov     bx,ds                   ;AN000; (tm11)
32724 0000219B 09DB                        or      bx,bx                   ;AN000; (tm11)
32725 0000219D 07                          pop     es                      ;AN000;
32726 0000219E 5D                          pop     bp                      ;AN000;
32727 0000219F 5F                          pop     di                      ;AN000;
32728 000021A0 5A                          pop     dx                      ;AN000;
32729 000021A1 59                          pop     cx                      ;AN000;
32730 000021A2 1F                          pop     ds                      ;AN000; (tm11)
32731 000021A3 58                          pop     ax                      ;AN000;
32732 000021A4 7424                        jz      short _$P_NON_DBCS      ;AN000;
32733                                  _$P_DBCS02:                           ;AN000;
32734 000021A6 2E8936[7819]                mov     [cs:_$P_DBCSEV_OFF],si  ;AC034; save EV offset
32735 000021AB 2E891E[7A19]                mov     [cs:_$P_DBCSEV_SEG],bx  ;AC034; save EV segment (tm11)
32736                                  _$P_DBCS00:                           ;AN000;
32737                                      ;mov    si,[cs:_$P_DBCSEV_OFF]  ;AC034; load EV offset
32738                                      ;mov    ds,[cs:_$P_DBCSEV_SEG]  ;AC034; and segment
32739                                      ; 08/07/2023
32740 000021B0 2EC536[7819]                lds     si,[cs:_$P_DBCSEV_OFF]
32741                                  _$P_DBCS_LOOP:                                ;AN000;
32742 000021B5 833C00                      cmp     word [si],0             ;AN000; zero vector ?
32743 000021B8 7410                        je      short _$P_NON_DBCS      ;AN000; then exit
32744 000021BA 3A04                        cmp     al,[si]                 ;AN000;
32745 000021BC 7208                        jb      short _$P_DBCS01        ;AN000; Check if AL is in
32746 000021BE 3A4401                      cmp     al,[si+1]               ;AN000;   range of
32747 000021C1 7703                        ja      short _$P_DBCS01        ;AN000;      the vector
32748 000021C3 F9                          stc                             ;AN000; if yes, indicate DBCS and exit
32749 000021C4 EB04                        jmp     short _$P_DBCS_EXIT     ;AN000;
32750                                  _$P_DBCS01:                           ;AN000;
32751 000021C6 46                          inc     si                      ;AC035; add '2' to
32752 000021C7 46                          inc     si                      ;AC035;  SI reg
32753                                                                      ;AN000; get next vector
32754 000021C8 EBEB                        jmp     short _$P_DBCS_LOOP     ;AN000; loop until zero vector found
32755                                  _$P_NON_DBCS:                                 ;AN000;
32756                                      ; 12/12/2022
32757                                      ; cf=0
32758                                      ;clc                            ;AN000; indicate SBCS
32759                                  _$P_DBCS_EXIT:                                ;AN000;
32760 000021CA 5B                          pop     bx                      ;AN000; (tm11)
32761 000021CB 5E                          pop     si                      ;AN000;
32762 000021CC 1F                          pop     ds                      ;AN000;
32763 000021CD C3                          retn                            ;AN000;
32764
32765                                  ; SYSCONF.ASM - MSDOS 6.0 - 1991
32766                                  ; ============================================================================
32767                                  ; 27/03/2019 - Retro DOS v4.0
32768
32769                                  ;control block definitions for parser.
32770                                  ;---------------------------------------------------------------------------
32771                                  ; buffer = [n | n,m] {/e}
32772
32773                                  ; 30/03/2019
32774
32775                                  struc p_parms
32776 00000000 ????                        resw    1       ; dw ?
32777 00000002 ??                          resb    1       ; db 1  ; an extra delimiter list
32778 00000003 ??                          resb    1       ; db 1 ; length is 1
32779 00000004 ??                          resb    1       ; db ';' ; delimiter
32780                                  .size:
32781                                  endstruc
32782
32783                                  struc p_pos
32784 00000000 ????                        resw    1       ; dw ?  ; numeric value??
32785 00000002 ????                        resw    1       ; dw ?  ; function
32786 00000004 ????                        resw    1       ; dw ?  ; result value buffer
32787
32788                                  ; note: by defining result_val before this structure, we could remove
32789                                  ;  the "result_val" from every structure invocation
32790
32791 00000006 ????                        resw    1       ; dw ?  ; value list
32792 00000008 ??                          resb    1       ; db 0  ; no switches/keywords
32793                                  .size:
32794                                  endstruc
32795
32796                                  struc       p_range
32797 00000000 ??                          resb    1       ; db 1  ; range definition
32798 00000001 ??                          resb    1       ; db 1  ; 1 definition of range
32799 00000002 ??                          resb    1       ; db 1  ; item tag for this range
32800 00000003 ????????                    resd    1       ; dd ?  ; numeric min
32801 00000007 ????????                    resd    1       ; dd ?  ; numeric max
32802                                  .size:
32803                                  endstruc
32804
32805                                  ;---------------------------------------------------------------------------
32806
32807                                      ; 26/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
32808                                      ; (SYSINIT:1F48h)
32809
32810                                      ; 08/07/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
32811                                      ; MSDOS 6.21 IO.SYS - SYSINIT:2083h
32812                                      ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:251Dh) ; (Retro DOS v5.0)
32813
32814                                  ; buffer = [n | n,m] {/e}
32815
32816                                  ;buf_parms p_parms <buf_parmsx>
32817                                  buf_parms:
32818 000021CE [D321]                      dw      buf_parmsx
32819 000021D0 01                          db      1       ; an extra delimiter list
32820 000021D1 01                          db      1       ; length is 1
32821 000021D2 3B                          db      ';'     ; delimiter
32822
32823                                  buf_parmsx:
32824 000021D3 0102[DD21][F121]            dw      201h,buf_pos1,buf_pos2; min 1, max 2 positionals
32825 000021D9 01                          db      1                     ; one switch
32826 000021DA [0522]                      dw      sw_x_ctrl
32827 000021DC 00                          db      0                     ; no keywords
32828
32829                                  ;buf_pos1 p_pos <8000h,0,result_val,buf_range_1>  ; numeric
32830                                  buf_pos1:
32831 000021DD 0080                        dw      8000h   ; numeric value??
32832 000021DF 0000                        dw      0       ; function
32833 000021E1 [1722]                      dw      result_val ; result value buffer
```

```
32834 000021E3 [E621]                  dw      buf_range_1 ; value list
32835 000021E5 00                      db      0       ; no switches/keywords
32836
32837                                  ;buf_range_1 p_range <,,,1,99>           ; M050
32838                                  buf_range_1:
32839 000021E6 01                      db      1       ; range definition
32840 000021E7 01                      db      1       ; 1 definition of range
32841 000021E8 01                      db      1       ; item tag for this range
32842 000021E9 01000000                dd      1       ; numeric min
32843 000021ED 63000000                dd      99      ; numeric max
32844
32845                                  ;buf_pos2 p_pos <8001h,0,result_val,buf_range_2> ; optional num.
32846                                  buf_pos2:
32847 000021F1 0180                    dw      8001h
32848 000021F3 0000                    dw      0
32849 000021F5 [1722]                  dw      result_val
32850 000021F7 [FA21]                  dw      buf_range_2
32851 000021F9 00                      db      0
32852
32853                                  ;buf_range_2 p_range <,,,0,8>
32854                                  buf_range_2:
32855 000021FA 01                      db      1
32856 000021FB 01                      db      1
32857 000021FC 01                      db      1
32858 000021FD 00000000                dd      0
32859 00002201 08000000                dd      8
32860
32861                                  ;sw_x_ctrl p_pos <0,0,result_val,noval,1> ; followed by one switch
32862                                  sw_x_ctrl:
32863 00002205 0000                    dw      0
32864 00002207 0000                    dw      0
32865 00002209 [1722]                  dw      result_val
32866 0000220B [1622]                  dw      noval
32867 0000220D 01                      db      1       ; 1 switch
32868
32869                                  switch_x:
32870 0000220E 2F5800                  db      '/X',0          ; M016
32871
32872                                  p_buffers:
32873 00002211 0000                    dw      0       ; local variables
32874                                  p_h_buffers:
32875 00002213 0000                    dw      0
32876                                  ; 26/10/2022  (MSDOS 5.0 IO.SYS SYSINIT compatibility)
32877                                  p_buffer_slash_x:
32878 00002215 00                      db      0 ; 31/03/2019
32879
32880                                  ;-- common definitions ---------------------------------------------
32881
32882 00002216 00                      noval:      db      0
32883
32884                                  result_val:         ;label  byte
32885 00002217 00                      db      0               ; type returned
32886                                  result_val_itag:
32887 00002218 00                      db      0               ; item tag returned
32888                                  result_val_swoff:
32889 00002219 0000                    dw      0               ; es:offset of the switch defined
32890                                  rv_byte:    ;label  byte
32891 0000221B 00000000                rv_dword: dd        0               ; value if number,or seg:offset to string.
32892
32893                                  ;-------------------------------------------------------------------------
32894
32895                                      ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
32896                                      ; (SYSINIT:1F99h)
32897
32898                                  ; break = [ on | off ]
32899
32900                                  ;brk_parms p_parms  <brk_parmsx>
32901                                  brk_parms:
32902 0000221F [2422]                    dw      brk_parmsx
32903 00002221 01                      db      1       ; an extra delimiter list
32904 00002222 01                      db      1       ; length is 1
32905 00002223 3B                      db      ';'     ; delimiter
32906
32907                                  brk_parmsx:
32908 00002224 0101[2A22]              dw      101h,brk_pos   ; min,max = 1 positional
32909 00002228 00                      db      0               ; no switches
32910 00002229 00                      db      0               ; no keywords
32911
32912                                  ;brk_pos p_pos <2000h,0,result_val,on_off_string> ; simple string
32913                                  brk_pos:
32914 0000222A 0020                    dw      2000h
32915 0000222C 0000                    dw      0
32916 0000222E [1722]                  dw      result_val
32917 00002230 [3322]                  dw      on_off_string
32918 00002232 00                      db      0
32919
32920                                  on_off_string:      ;label  byte
32921 00002233 03                      db      3               ; signals that there is a string choice
32922 00002234 00                      db      0               ; no range definition
32923 00002235 00                      db      0               ; no numeric values choice
32924 00002236 02                      db      2               ; 2 strings for choice
32925 00002237 01                      db      1               ; the 1st string tag
32926 00002238 [3D22]                  dw      on_string
32927 0000223A 02                      db      2               ; the 2nd string tag
32928 0000223B [4022]                  dw      off_string
32929
32930                                  on_string:
32931 0000223D 4F4E00                  db      "ON",0
32932                                  off_string:
32933 00002240 4F464600                db      "OFF",0
32934
32935                                  p_ctrl_break:
32936 00002244 00                      db      0       ; local variable
32937
32938                                  ;-------------------------------------------------------------------------
32939
32940                                      ; 27/10/2022
32941
32942                                  ; country = n {m {path}}
32943                                  ; or
32944                                  ; country = n,,path
32945
32946                                  ;cntry_parms p_parms <cntry_parmsx>
32947                                  cntry_parms:
32948 00002245 [4A22]                  dw      cntry_parmsx
32949 00002247 01                      db      1
32950 00002248 01                      db      1
32951 00002249 3B                      db      ';'
32952
32953                                  cntry_parmsx:
32954 0000224A 0103[5422][6822]-       dw      301h,cntry_pos1,cntry_pos2,cntry_pos3 ; min 1, max 3 pos.
32954 00002250 [7122]
32955 00002252 00                      db      0               ; no switches
32956 00002253 00                      db      0               ; no keywords
```

```
                                ;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
```

```
                                
000022D3 00                     p_fcbs:     db      0               ; local variable
000022D4 00                     p_keep:     db      0               ; local variable
                                
                                ;--------------------------------------------------------------------------
                                
                                    ; 27/10/2022
                                
                                ; lastdrive = x
                                
                                ;ldrv_parms p_parms <ldrv_parmsx>
                                ldrv_parms:
000022D5 [DA22]                     dw      ldrv_parmsx
000022D7 01                         db      1
000022D8 01                         db      1
000022D9 3B                         db      ';'
                                
                                ldrv_parmsx:
000022DA 0101[E022]                 dw      101h,ldrv_pos  ; min,max = 1 positional
000022DE 00                         db      0              ; no switches
000022DF 00                         db      0              ; no keywords
                                
                                ;ldrv_pos p_pos    <110h,10h,result_val,noval> ; drive only, ignore colon
                                ldrv_pos:                               ; remove colon at end
000022E0 1001                       dw      110h
000022E2 1000                       dw      10h
000022E4 [1722]                     dw      result_val
000022E6 [1622]                     dw      noval
000022E8 00                         db      0
                                
000022E9 00                     p_ldrv:     db      0               ; local variable
                                
                                ;--------------------------------------------------------------------------
                                
                                    ; 27/10/2022
                                
                                ; stacks = n,m
                                
                                ;stks_parms p_parms <stks_parmsx>
                                stks_parms:
000022EA [EF22]                     dw      stks_parmsx
000022EC 01                         db      1
000022ED 01                         db      1
000022EE 3B                         db      ';'
                                
                                stks_parmsx:
000022EF 0202[F722][0B23]           dw      202h,stks_pos_1,stks_pos_2 ; min,max = 2 positionals
000022F5 00                         db      0               ; no switches
000022F6 00                         db      0               ; no keywords
                                
                                ;stks_pos_1 p_pos <8000h,0,result_val,stks_range> ; numeric value
                                stks_pos_1:
000022F7 0080                       dw      8000h
000022F9 0000                       dw      0
000022FB [1722]                     dw      result_val
000022FD [0023]                     dw      stks_range
000022FF 00                         db      0
                                
                                ;stks_range p_range <,,,0,64>
                                stks_range:
00002300 01                         db      1
00002301 01                         db      1
00002302 01                         db      1
00002303 00000000                   dd      0
00002307 40000000                   dd      64
                                
                                ;stks_pos_2 p_pos <8000h,0,result_val,stk_size_range> ; numeric value
                                stks_pos_2:
0000230B 0080                       dw      8000h
0000230D 0000                       dw      0
0000230F [1722]                     dw      result_val
00002311 [1423]                     dw      stk_size_range
00002313 00                         db      0
                                
                                ;stk_size_range p_range <,,,0,512>
                                stk_size_range:
00002314 01                         db      1
00002315 01                         db      1
00002316 01                         db      1
00002317 00000000                   dd      0
0000231B 00020000                   dd      512
                                
                                p_stack_count:
0000231F 0000                       dw      0       ; local variable
                                p_stack_size:
00002321 0000                       dw      0       ; local variable
                                
                                ;--------------------------------------------------------------------------
                                
                                    ; 27/10/2022
                                
                                ; multitrack = [ on | off ]
                                
                                ;mtrk_parms p_parms <mtrk_parmsx>
                                mtrk_parms:
00002323 [2823]                     dw      mtrk_parmsx
00002325 01                         db      1
00002326 01                         db      1
00002327 3B                         db      ';'
                                
                                mtrk_parmsx:
00002328 0101[2E23]                 dw      101h,mtrk_pos  ; min,max = 1 positional
0000232C 00                         db      0              ; no switches
0000232D 00                         db      0              ; no keywords
                                
                                ;mtrk_pos p_pos <2000h,0,result_val,on_off_string> ; simple string
                                mtrk_pos:
0000232E 0020                       dw      2000h
00002330 0000                       dw      0
00002332 [1722]                     dw      result_val
00002334 [3322]                     dw      on_off_string
00002336 00                         db      0
                                
00002337 00                     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:
```

```
33205 00002338 [3D23]                      dw      swit_parmsx
33206 0000233A 01                          db      1
33207 0000233B 01                          db      1
33208 0000233C 3B                          db      ';'
33209
33210                                  swit_parmsx:
33211 0000233D 0000                        dw      0               ; no positionals
33212                                      ; 08/07/2023 - Retro DOS v4.2 (MSDOS 6.21 IO.SYS, SYSINIT)
33213                                      ;db     5                 ; # of switches
33214                                      ; 14/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM, SYSINIT)
33215 0000233F 06                          db      6
33216                                      ; 27/10/2022 - Retro DOS v4.0 (MSDOS 5.0 IO.SYS, SYSINIT)
33217                                      ;db     3
33218 00002340 [4D23]                      dw      swit_k_ctrl     ; /k control
33219                                      ; 01/01/2023 - Retro DOS v4.2 ; *
33220 00002342 [5923]                      dw      swit_n_ctrl ; * ; /n control (for MULTI_CONFIG only)
33221 00002344 [6523]                      dw      swit_f_ctrl ; * ; /f control (for MULTI_CONFIG only)
33222 00002346 [7123]                      dw      swit_t_ctrl     ; /t control
33223 00002348 [7D23]                      dw      swit_w_ctrl     ; /w control
33224                                      ; 14/04/2024 - Retro DOS v5.0 ; **
33225 0000234A [8923]                      dw      swit_i_ctrl     ; /i control
33226 0000234C 00                          db      0               ; no keywords
33227
33228                                  ;swit_k_ctrl p_pos <0,0,result_val,noval,1> ; switch string follows
33229                                  swit_k_ctrl:
33230 0000234D 00000000[1722]-             dw      0,0,result_val,noval
33230 00002353 [1622]
33231 00002355 01                          db      1
33232 00002356 2F4B00                  swit_k:     db      '/K',0
33233
33234                                  ; 01/01/2023 - Retro DOS v4.2 (MSDOS 6.21 IO.SYS)
33235                                  ; (SYSINIT:220Ch) ; *
33236
33237                                  ; 27/10/2022 - Retro DOS v4.0 (MSDOS 5.0 IO.SYS, SYSINIT)
33238                                  ;
33239                                  ;swit_n_ctrl p_pos <0,0,result_val,noval,1> ; switch string follows
33240                                  swit_n_ctrl: ; *
33241 00002359 00000000[1722]-             dw      0,0,result_val,noval
33241 0000235F [1622]
33242 00002361 01                          db      1
33243 00002362 2F4E00                  swit_n: db  '/N',0
33244
33245                                  ;swit_f_ctrl p_pos <0,0,result_val,noval,1> ; switch string follows
33246                                  swit_f_ctrl: ; *
33247 00002365 00000000[1722]-             dw      0,0,result_val,noval
33247 0000236B [1622]
33248 0000236D 01                          db      1
33249 0000236E 2F4600                  swit_f: db  '/F',0
33250
33251                                      ; 27/10/2022
33252
33253                                  ;swit_t_ctrl p_pos <0,0,result_val,noval,1> ; switch string follows    M059
33254                                  swit_t_ctrl:
33255 00002371 00000000[1722]-             dw      0,0,result_val,noval
33255 00002377 [1622]
33256 00002379 01                          db      1
33257 0000237A 2F5400                  swit_t:     db      '/T',0                  ;                           M059
33258                                  ;swit_w_ctrl p_pos <0,0,result_val,noval,1> ; switch string follows    M063
33259                                  swit_w_ctrl:
33260 0000237D 00000000[1722]-             dw      0,0,result_val,noval
33260 00002383 [1622]
33261 00002385 01                          db      1
33262 00002386 2F5700                  swit_w:     db      '/W',0                  ;                           M063
33263
33264                                  ; 14/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM, SYSINIT)
33265                                  ;;;
33266                                  ;swit_i_ctrl p_pos <0,0,result_val,noval,1> ; switch string follows
33267                                  swit_i_ctrl:
33268 00002389 0000                        dw      0
33269 0000238B 0000                        dw      0
33270 0000238D [1722]                      dw      result_val
33271 0000238F [1622]                      dw      noval
33272 00002391 01                          db      1
33273 00002392 2F4900                  swit_i:     db      '/I',0
33274                                  ;;;
33275
33276                                  ;   There doesn't need to be p_swit_n or p_swit_f because /N and /F are
33277                                  ;   acted upon during MULTI_CONFIG processing; we only needed entries
33278                                  ;   in the above table to prevent the parsing code from complaining about them
33279
33280 00002395 00                      p_swit_k:   db      0       ; local variable
33281 00002396 00                      p_swit_t:   db      0       ; local variable                    M059
33282 00002397 00                      p_swit_w:   db      0       ; local variable                    M063
33283                                  ; 14/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
33284 00002398 00                      p_swit_i:   db      0
33285
33286                                  ;---------------------------------------------------------------------------
33287
33288                                      ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
33289                                      ; (SYSINIT:20E8h)
33290
33291                                  ; DOS = [ high | low ]
33292
33293                                  ;dos_parms p_parms  <dos_parmsx>
33294                                  dos_parms:
33295 00002399 [9E23]                      dw      dos_parmsx
33296 0000239B 01                          db      1
33297 0000239C 01                          db      1
33298 0000239D 3B                          db      ';'
33299                                  dos_parmsx:
33300 0000239E 01                          db      1               ; min parameters
33301 0000239F 02                          db      2               ; max parameters
33302 000023A0 [A623]                      dw      dos_pos         ;
33303 000023A2 [A623]                      dw      dos_pos         ;
33304 000023A4 00                          db      0               ; no switches
33305 000023A5 00                          db      0               ; no keywords
33306
33307                                  ;dos_pos p_pos     <2000h,0,result_val,dos_strings> ; simple string
33308                                  ;        p_pos     <2000h,0,result_val,dos_strings> ; simple string
33309                                  dos_pos:
33310 000023A6 00200000[1722]-             dw      2000h,0,result_val,dos_strings
33310 000023AC [B823]
33311 000023AE 00                          db      0
33312 000023AF 00200000[1722]-             dw      2000h,0,result_val,dos_strings
33312 000023B5 [B823]
33313 000023B7 00                          db      0
33314
33315                                  dos_strings:        ;label  byte
33316 000023B8 03                          db      3               ; signals that there is a string choice
33317 000023B9 00                          db      0               ; no range definition
33318 000023BA 00                          db      0               ; no numeric values choice
33319 000023BB 04                          db      4               ; 4 strings for choice
33320 000023BC 01                          db      1               ; the 1st string tag
33321 000023BD [E623]                      dw      hi_string
```

```
33322 000023BF 02                       db      2               ; the 2nd string tag
33323 000023C0 [EB23]                   dw      lo_string
33324 000023C2 03                       db      3
33325 000023C3 [EF23]                   dw      umb_string
33326 000023C5 04                       db      4
33327 000023C6 [F323]                   dw      noumb_string
33328
33329                                   ; 14/04/2024 - Retro DOS v5.0
33330                                   ; (PCDOS 7.1 IBMDOS.COM - SYSINIT:273Eh)
33331                                   ;;;
33332                                   dosdata_parms:
33333 000023C8 [CD23]                       dw      dosdata_parmsx ; DOSDATA = UMB|NOUMB
33334 000023CA 01                           db      1
33335 000023CB 01                           db      1
33336 000023CC 3B                           db      ';'
33337                                   dosdata_parmsx:
33338 000023CD 01                           db      1
33339 000023CE 01                           db      1               ; min,max = 1 positional
33340 000023CF [D323]                       dw      dosdata_pos
33341 000023D1 00                           db      0               ; no switches
33342 000023D2 00                           db      0               ; no keywords
33343
33344                                       ; dosdata_pos p_pos <2000h,0,result_val,dosdata_strings>
33345                                   dosdata_pos:
33346 000023D3 0020                         dw      2000h           ; simple string
33347 000023D5 0000                         dw      0
33348 000023D7 [1722]                       dw      result_val
33349 000023D9 [DC23]                       dw      dosdata_strings
33350 000023DB 00                           db      0
33351                                   dosdata_strings:
33352 000023DC 03                           db      3               ; signals that there is a string choice
33353 000023DD 00                           db      0               ; no range definition
33354 000023DE 00                           db      0               ; no numeric values choice
33355 000023DF 02                           db      2               ; 2 strings for choice
33356 000023E0 01                           db      1               ; the 1st string tag
33357 000023E1 [EF23]                       dw      umb_string      ; "UMB"
33358 000023E3 02                           db      2               ; the 2nd string tag
33359 000023E4 [F323]                       dw      noumb_string    ; "NOUMB"
33360                                   ;;;
33361
33362 000023E6 4849474800               hi_string:  db      "HIGH",0
33363 000023EB 4C4F5700                 lo_string:  db      "LOW",0
33364 000023EF 554D4200                 umb_string: db      "UMB",0
33365 000023F3 4E4F554D4200             noumb_string:       db      "NOUMB",0
33366
33367                                   p_dos_hi:
33368 000023F9 00                           db      0       ; local variable
33369                                                       ; BUGBUG : I dont know whether PARSER uses
33370                                                       ;          this variable or not
33371                                       ; 14/04/2024 (PCDOS 7.1 IBMBIO.COM)
33372 000023FA 00                           db      0
33373
33374                                   ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
33375                                   ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
33376                                   ;%if 0
33377
33378                                   ;************************************************************************ RICHID ****
33379
33380                                   ;include    highvar.inc    ; devicehigh variables (used by loadhigh also)
33381
33382                                   ; 30/03/2019 - Retro DOS v4.0
33383                                   ;--------------------------------------------------------------------------------
33384
33385                                   ;   Module:   HIGHVAR.INC - Data common to LOADHIGH and DEVICEHIGH, res seg
33386                                   ;
33387                                   ;   Date:     May 14, 1992
33388                                   ;
33389                                   ;*******************************************************************************
33390                                   ;
33391                                   ;   Modification log:
33392                                   ;
33393                                   ;     DATE    WHO      DESCRIPTION
33394                                   ;   --------  -------  -------------------------------------------------------
33395                                   ;   05/14/92  t-richj  Original
33396                                   ;   06/21/92  t-richj  Final revisions before check-in
33397                                   ;
33398                                   ;*******************************************************************************
33399                                   ;
33400                                   ; There are two primary definitions which need to be made, selectively, before
33401                                   ; this include file should be used.  These are:
33402                                   ;    HV_Extern - If this has been defined, variables for this module will be
33403                                   ;                declared as external.  Otherwise, variables will be declared
33404                                   ;                public, as well as defined, here.  LoadHigh declares HV_Extern
33405                                   ;                in stub.asm and loadhi.asm, and does not declare it in
33406                                   ;                rdata.asm... DeviceHigh does not declare HV_Extern anywhere
33407                                   ;                (as only one module, sysconf.asm, includes this file).
33408                                   ;    HV_LoadHigh - This should be defined when this module is going into
33409                                   ;                  command.com, for LoadHigh.  All of loadhi.asm, stub.asm and
33410                                   ;                  rdata.asm define this, while io.sys' sysconf.asm does not.
33411                                   ;
33412                                   ;*******************************************************************************
33413
33414                                   ; To keep track of which UMBs were specified on the DH/LH command lines, and
33415                                   ; to keep track of the minimum sizes given for each, there're two arrays kept
33416                                   ; in { IO.SYS: sysinitseg / COMMAND.COM: DATARES }... each is MAXUMB elements
33417                                   ; big.  16 should be around 14 too many for most users, so there's no expected
33418                                   ; space problem (it's just such a nice round number, eh?).
33419
33420                                   MAXUMB      equ     16
33421
33422                                   ; Memory elements owned by the system are marked as PSP address 8 in both the
33423                                   ; USA and Japan; Japanese systems also use 9 under more bizzarre conditions.
33424
33425                                   FreePSPOwner        equ     0       ; Free MCBs all have an owner PSP address of 0
33426                                   SystemPSPOwner      equ     8
33427                                   ;JapanPSPOwner      equ     9
33428
33429                                   ; for LoadHigh and DeviceHigh:
33430                                   ;
33431                                   ;   fInHigh - Is set to 1 during HideUMBs(), and back to zero in
33432                                   ;             UnHideUMBs().
33433                                   ;   fUmbTiny - Is set to 1 iff the user has specified /S on the command-
33434                                   ;              line.
33435                                   ;   SegLoad - Segment address for first UMB specified; set automatically.
33436                                   ;   UmbLoad - The load UMB number; for example, this is 3 if the user has
33437                                   ;             given a command-line like "/L:3,500;4"
33438                                   ;   UmbUsed - An array of characters, each of which is 1 iff the UMB
33439                                   ;             matching its index number was specified on the command-line;
33440                                   ;             for example, after "/L:3,500;4;7", UmbUsed[3], [4] and [7]
33441                                   ;             will be set to 1.  All others will be set to 0.
33442                                   ;   UmbSize - An array of words, each of which is interpereted as a size
33443                                   ;             specified by the user for a UMB (in the above example, all
33444                                   ;             elements would be zero save UmbSize[3], which would be 500.
33445                                   ;   fm_umb - Set to the old UMB link-state (0x80 or 0x00)
```

```
33446                                   ;   fm_strat - Set to the old memory-allocation strategy (0$00000???)
33447                                   ;   fm_argc  - Number of arguments received by ParseVar() (see ParseVar()
33448                                   ;              for details).
33449
33450 000023FB 00                       fInHigh:  db       0
33451 000023FC 00                       fUmbTiny: db       0
33452 000023FD 0000                     SegLoad:  dw       0
33453 000023FF 00                       UmbLoad:  db       0
33454 00002400 00<rep 10h>              UmbUsed:  times MAXUMB db 0 ; times 16 db 0  ; db 16 dup(?)
33455 00002410 0000<rep 10h>            UmbSize:  times MAXUMB dw 0 ; times 16 dw 0  ; dw 16 dup(?)
33456 00002430 00                       fm_umb:   db       0
33457 00002431 00                       fm_strat: db       0
33458 00002432 00                       fm_argc:  db       0
33459
33460                                   ; UmbLoad is set to UNSPECIFED, below, until /L:umb is read; at which point
33461                                   ; UmbLoad is set to the UMB number given.
33462
33463                                   UNSPECIFIED equ    -1
33464
33465                                   ;%endif ; 27/10/2022
33466
33467                                   ;******************************************************************** RICHID ****
33468
33469                                   ; 30/03/2019 - Retro DOS v4.0 (MSDOS 6.0, SYSCONF.ASM)
33470                                   ; ((MSDOS 6.21 IO.SYS -> SYNINIT:22BAh))
33471
33472                                   ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
33473                                   ; (SYSINIT:212Bh)
33474
33475                                               ;public DevEntry
33476
33477 00002433 0000                     DevSize:    dw     0      ; size of the device driver being loaded(paras)
33478 00002435 0000                     DevLoadAddr:       dw     0      ; Mem addr where the device driver is 2 b loaded
33479 00002437 0000                     DevLoadEnd: dw     0      ; MaxAddr to which device can be loaded
33480 00002439 00000000                 DevEntry:   dd     0      ; Entry point to the device driver
33481 0000243D 00000000                 DevBrkAddr: dd     0      ; Break address of the device driver
33482                                   ; 30/12/2022
33483                                   ; 27/10/2022
33484 00002441 00                       ConvLoad:   db     0      ; Use conventional (dos 5 -style) InitDevLoad?
33485                                   ;
33486 00002442 00                       DevUMB:            db     0      ; byte indicating whether to load DDs in UMBs
33487 00002443 0000                     DevUMBAddr: dw     0      ; cuurent UMB used fro loading devices (paras)
33488 00002445 0000                     DevUMBSize: dw     0      ; Size of the current UMB being used   (paras)
33489 00002447 0000                     DevUMBFree: dw     0      ; Start of free are in the current UMB (paras)
33490                                   ;
33491 00002449 00000000                 DevXMSAddr: dd     0
33492                                   ;
33493 0000244D 0000                     DevExecAddr:       dw     0      ; Device load address parameter to Exec call
33494 0000244F 0000                     DevExecReloc:      dw     0      ; Device load relocation factor
33495                                   ;
33496 00002451 00                       DeviceHi:   db     0      ; Flag indicating whther the current device
33497                                                              ;  is being loaded into UMB
33498 00002452 0000                     DevSizeOption:     dw     0      ; SIZE= option
33499                                   ;
33500 00002454 00                       Int12Lied:  db     0      ; did we trap int 12 ?
33501 00002455 0000                     OldInt12Mem:       dw     0      ; value in 40:13h (int 12 ram)
33502 00002457 50524F544D414E24         ThreeComName:      db     'PROTMAN$'     ; 3Com Device name
33503                                   ;
33504 0000245F 00                       FirstUMBLinked:    db     0
33505 00002460 0000                     DevDOSData: dw     0      ; segment of DOS Data
33506 00002462 00000000                 DevCmdLine: dd     0      ; Current Command line
33507 00002466 00                       DevSavedDelim:     db     0      ; The delimiter which was replaced with null
33508                                                              ; to use the file name in the command line
33509                                   ; 13/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
33510                                   ;   ifdef  dblspace_hooks
33511 00002467 00                       MagicHomeFlag:     db     0      ; set non-zero when MagicDrv is final placed
33512                                   ;   endif
33513
33514                                   ; ==========================================================================
33515
33516                                   ; 31/03/2019 - Retro DOS v4.0
33517
33518                                   ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
33519                                   ; (SYSINIT:215Eh)
33520
33521                                   ;--------------------------------------------------------------------------
33522                                   ;
33523                                   ; procedure : doconf
33524                                   ;
33525                                   ;             Config file is parsed initially with this routine. For the
33526                                   ;             Subsequent passes 'multi_pass' entry is used .
33527                                   ;
33528                                   ;--------------------------------------------------------------------------
33529
33530                                       ; 27/10/2022
33531                                   doconf:
33532 00002468 0E                           push    cs
33533 00002469 1F                           pop     ds
33534
33535 0000246A B80037                       mov     ax,3700h
33536                                           ;mov       ax,(CHAR_OPER<<8)      ; get switch character
33537 0000246D CD21                         int     21h
33538 0000246F 8816[BC4B]                   mov     [command_line+1],dl    ; set in default command line
33539
33540                                   ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21  IO.SYS)
33541                                   ; 27/10/2022
33542                                   ;;ifdef     MULTI_CONFIG
33543                                   ;   ;mov    [command_line-1],dl     ; save default switchchar
33544 00002473 8816[BA4B]                   mov     [def_swchr],dl ; 31/03/2019
33545                                   ;;endif     ;MULTI_CONFIG
33546
33547 00002477 BA[D14A]                     mov     dx,config ;'\CONFIG.SYS' ;now pointing to file description
33548 0000247A B8003D                       mov     ax,3D00h
33549                                       ;mov    ax,OPEN<<8             ;open file "config.sys"
33550 0000247D F9                           stc                            ;in case of int 24
33551 0000247E CD21                         int     21h                    ;function request
33552 00002480 7309                         jnc     short noprob           ; brif opened okay
33553
33554                                   ; 31/12/2022
33555                                   ; 27/10/2022
33556                                   ;;ifdef     MULTI_CONFIG
33557 00002482 E8A019                       call    kbd_read               ; we still want to give the guy
33558                                   ;                                  ; a chance to select clean boot!
33559                                   ;;endif                                 ; (ie, no autoexec.bat processing)
33560 00002485 C606[CD02]0B                 mov     byte [multi_pass_id],11        ; set it to unreasonable number
33561 0000248A C3                           retn
33562                                   noprob:                            ;get file size (note < 64k!!)
33563 0000248B 89C3                         mov     bx,ax  ; File handle
33564 0000248D 31C9                         xor     cx,cx                  ; 0
33565 0000248F 31D2                         xor     dx,dx                  ; 0
33566                                       ;mov    ax,4202h
33567 00002491 B80242                       mov     ax,(LSEEK<<8)|2
33568 00002494 CD21                         int     21h
33569 00002496 A3[5603]                     mov     [count],ax             ; dx:ax = file size ; 08/09/2023
```

```
                                                        ; 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
00002503 C3                         retn
                                ;--------------------------------------------------------------------------
                                ;
                                ; entry : multi_pass
                                ;
                                ;             called to execute device=,install= commands
                                ;
                                ;--------------------------------------------------------------------------

                                    ; 27/10/2022
                                multi_pass:
00002504 0E                         push    cs
00002505 1F                         pop     ds

00002506 803E[CD02]0A               cmp     byte [multi_pass_id],10
                                ;jae_endconv:
0000250B 73F6                       jae     short endconv         ; do nothing. just return.

                                    ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
0000250D FF36[A302]                 push    word [CONFBOT]
                                    ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;push   word [top_of_cdss]
00002511 07                         pop     es                    ; es -> confbot

00002512 8B36[5803]                 mov     si,[org_count]
00002516 8936[5603]                 mov     [count],si            ; set count
0000251A 31F6                       xor     si,si ; 0
0000251C 8936[5A03]                     mov     [chrptr],si             ; reset chrptr
00002520 8936[AF02]                     mov     [linecount],si          ; reset linecount

00002524 E88822                     call    getchr
00002527 EB06                       jmp     short conflp

                                    ; 14/04/2024
                                    ; 01/01/2023
                                ;endconv:
                                    ;retn

                                getcom:
                                    ; 03/01/2023
                                    ; ds = cs
0000252A E8C016                         call    organize                ; organize the file
0000252D E88022                     call    getchr
                                conflp:
00002530 72D2                       jc      short endconv

00002532 FF06[AF02]                     inc     word [linecount]  ; increase linecount

                                    ; 08/09/2023
00002536 30E4                       xor     ah,ah ; 0
                                    ;mov    byte [multdeviceflag],0       ; reset multdeviceflag.
                                    ;mov    byte [setdevmarkflag],0       ; reset setdevmarkflag.
00002538 8826[6619]                 mov     [multdeviceflag],ah ; 0
0000253C 8826[6919]                 mov     [setdevmarkflag],ah ; 0

00002540 3C0A                       cmp     al,lf                 ; linefeed?
00002542 7448                       je      short blank_line      ;  then ignore this line.

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:23CCh)
                                ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0

                                ;ifdef      MULTI_CONFIG

                                ;   If this is a genuine CONFIG.SYS command, then there should be a line
                                ;   number immediately following it....

00002544 A2[6419]                       mov     [config_cmd],al         ; save original command code
                                    ;and    al,NOT CONFIG_OPTION_QUERY
00002547 247F                       and     al,~CONFIG_OPTION_QUERY ; and al,7Fh

                                    ; 08/09/2023
00002549 3826[6519]                 cmp     [config_multi],ah ; 0
                                    ;cmp    byte [config_multi],0 ; is this a multi-config config.sys?
0000254D 7427                           je      short not_final         ; no, line number is not embedded

0000254F 50                             push    ax                      ;
00002550 E85D22                         call    getchr                  ; ignore end-of-image errors,
00002553 88C4                           mov     ah,al                   ; because if there's an error
00002555 E85822                         call    getchr                  ; fetching the line number that's
00002558 86E0                           xchg    al,ah                   ; supposed to be there, the next
0000255A A3[AF02]                       mov     [linecount],ax          ; getchr call will get the same error
0000255D 58                             pop     ax
                                ;
                                ;   HACK: when 4DOS.COM is the shell and it doesn't have an environment from
                                ;   which to obtain its original program name, it grovels through all of
                                ;   memory to find the filename that was used to exec it; it wants to find
                                ;   the SHELL= line in the in-memory copy of CONFIG.SYS, and it knows that
                                ;   sysinit converts the SHELL= keyword to an 'S', so it expects to find an 'S'
                                ;   immediately before the filename, but since we are now storing line # info
                                ;   in the config.sys memory image, 4DOS fails to find the 'S' in the right
                                ;   spot.
                                ;
                                ;   So, on the final pass of CONFIG.SYS, copy the command code (eg, 'S')
                                ;   over the line number info, since we no longer need that info anyway. This
                                ;   relies on the fact that getchr leaves ES:SI pointing to the last byte
                                ;   retrieved.
                                ;
0000255E 803E[CD02]02                   cmp byte [multi_pass_id],2; final pass?
00002563 7211                           jb  short not_final             ; no
                                        ;test      word [install_flag],have_install_cmd
00002565 F606[CE02]01               test    byte [install_flag],have_install_cmd ; 1
0000256A 7407                           jz  short final             ; no install cmds, so yes it is
0000256C 803E[CD02]03                   cmp byte [multi_pass_id],3; final pass?
00002571 7203                           jb  short not_final             ; no
                                final:                                  ;
00002573 268804                     mov     [es:si],al          ; save backward-compatible command code
                                not_final:                              ;
                                ;endif

                                ; 31/12/2022
                                ;%endif ; 27/10/2022

00002576 88C4                       mov     ah,al
00002578 E83522                     call    getchr
0000257B 7314                       jnc     short tryi

0000257D 803E[CD02]02               cmp     byte [multi_pass_id],2
                                    ;jae    short jae_endconv     ; do not show badop again for multi_pass.
                                    ; 27/10/2022
00002582 7380                       jnb     short endconv
```

```
                            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

000025E6 81CB8000                       or  bx,HIGH_FIRST  ; 80h  ;set alloc to HighFirst
000025EA B80158                     mov     ax,(ALLOCOPER<<8)|1 ; 5801h
000025ED CD21                       int     21h                     ;set alloc strategy

000025EF B80258                     mov      ax,(ALLOCOPER<<8)|2 ; 5802h
000025F2 CD21                           int     21h                     ; get link state
000025F4 30E4                           xor     ah,ah                   ; clear top byte
000025F6 50                             push    ax                      ; save for return

000025F7 B80358                         mov ax,(ALLOCOPER<<8)|3 ; 5803h
000025FA BB0100                     mov     bx,1
000025FD CD21                       int     21h                     ;link in UMBs

000025FF E897EC                     call    do_install_exec         ;install it.

00002602 B80358                         mov     ax,(ALLOCOPER<<8)|3
00002605 5B                             pop     bx                      ; recover original link state
00002606 CD21                           int     21h
00002608 5B                             pop     bx                      ; recover original alloc strategy
00002609 B80158                         mov     ax,(ALLOCOPER<<8)|1
0000260C CD21                           int     21h

                                    ;jmp    short coff              ;to handle next install= command.
                                    ; 01/01/2023
0000260E EBA7                       jmp     short multi_pass_coff2

                                ;%endif ; 27/10/2022

                                multi_pass_filter:
00002610 80FC59                         cmp     ah,CONFIG_COMMENT ; 'Y' ; comment?
00002613 740A                       je      short multi_pass_adjust
00002615 80FC5A                         cmp     ah,CONFIG_UNKNOWN ; 'Z' ; bad command?
00002618 7405                       je      short multi_pass_adjust
0000261A 80FC30                         cmp     ah,CONFIG_REM  ; '0'     ; rem?
0000261D 7508                       jne     short multi_pass_coff ; ignore the rest of the commands.

                                multi_pass_adjust:              ; these commands need to
0000261F FF0E[5A03]                     dec     word [chrptr]           ;  adjust chrptr,count
00002623 FF06[5603]                     inc     word [count]            ;  for newline proc.

                                multi_pass_coff:
                                    ; 11/12/2022
                                    ;jmp    short coff              ; to handle next install= commands.
                                    ; 01/01/2023
00002627 EB8E                       jmp     short multi_pass_coff2

                                ;--------------------------------------------------------------------------
                                ; buffer command
                                ;--------------------------------------------------------------------------

                                ;******************************************************************************
                                ;                                                                             *
                                ; function: parse the parameters of buffers= command.                        *
                                ;                                                                             *
                                ; input :                                                                     *
                                ;   es:si -> parameters in command line.                                      *
                                ; output:                                                                    *
                                ;   buffers set                                                             *
                                ;   buffer_slash_x flag set if /x option chosen.                            *
                                ;   h_buffers set if secondary buffer cache specified.                      *
                                ;                                                                           *
                                ; subroutines to be called:                                                  *
                                ;   sysinit_parse                                                           *
                                ; logic:                                                                    *
                                ; {                                                                         *
                                ;   set di points to buf_parms;  /*parse control definition*/              *
                                ;   set dx,cx to 0;                                                         *
                                ;   reset buffer_slash_x;                                                   *
                                ;   while (end of command line)                                             *
                                ;   { sysinit_parse;                                                        *
                                ;     if (no error) then                                                    *
                                ;          if (result_val._$P_synonym_ptr == slash_e) then /*not a switch *
                                ;              buffer_slash_x = 1                                         *
                                ;          else if  (cx == 1) then          /* first positional */    *
                                ;                   buffers = result_val._$P_picked_val;                  *
                                ;               else  h_buffers = result_val._$P_picked_val;              *
                                ;     else {show error message;error exit}                                  *
                                ;   };                                                                      *
                                ;   if (buffer_slash_x is off & buffers > 99) then show_error;              *
                                ; };                                                                        *
                                ;                                                                           *
                                ;******************************************************************************

                                    ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:229Ch)
                                tryb:
                                        cmp     ah,CONFIG_BUFFERS ; 'B'
00002629 80FC42                     jne     short tryc
0000262C 755C

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:24BFh)
                                ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                ;ifdef      MULTI_CONFIG
0000262E E8EB1F                     call    query_user              ; query the user if config_cmd
00002631 7257                       jc      short tryc              ; has the CONFIG_OPTION_QUERY bit set
                                ;endif
                                ;%endif ; 27/10/2022

                                    ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ; 18/12/2022
00002633 31C9                       xor     cx,cx
                                    ;mov    byte [p_buffer_slash_x],0 ; 31/03/2019
00002635 880E[1522]                 mov     [p_buffer_slash_x],cl ; 0

00002639 BF[CE21]                   mov     di,buf_parms
                                    ;xor    cx,cx   ; 18/12/2022
                                    ; 03/01/2023
                                    ;mov    dx,cx
                                do7:
0000263C E82808                     call    sysinit_parse
0000263F 7303                       jnc     short if7               ; parse error,
                                    ;call   badparm_p               ;  and show messages and end the search loop.
                                    ;;jmp   short sr7
                                    ; 31/12/2022
                                ;sr7:
                                    ;jmp    coff
                                    ; 03/01/2023
00002641 E91207                     jmp     badparm_p_coff
                                if7:
00002644 83F8FF                     cmp     ax,_$P_RC_EOL ; 0FFFFh; end of line?
00002647 741A                       je      short en7               ;  then jmp to $endloop for semantic check
```

```
                                        ;cmp    word [result_val_swoff],switch_x
00002649 813E[1922][0E22]               cmp     word [result_val+_$P_Result_Blk.SYNONYM_Ptr],switch_x
                                        ;jne    short if11
                                        ; 31/12/2022
0000264F 74EB                           je      short do7 ;je short en11

                                    ;   mov     byte [p_buffer_slash_x],1 ; set the flag M016
                                        ;jmp    short en11 ; 31/12/2022
                                    if11:
                                        ;mov    ax,[rv_dword]
00002651 A1[1B22]                       mov     ax,[result_val+_$P_Result_Blk.Picked_Val]
00002654 83F901                         cmp     cx,1
00002657 7505                           jne     short if13

00002659 A3[1122]                       mov     [p_buffers],ax
                                        ;jmp    short en11
                                        ; 31/12/2022
0000265C EBDE                           jmp     short do7
                                    if13:
0000265E A3[1322]                       mov     [p_h_buffers],ax
                                    en11:
00002661 EBD9                           jmp     short do7
                                    en7:
00002663 833E[1122]63                   cmp     word [p_buffers],99
00002668 760B                           jbe     short if18

                                    ;   cmp     byte [p_buffer_slash_x],0 ; M016
                                    ;   jne     short if18

0000266A E82508                         call    badparm_p
0000266D C706[1322]0000                 mov     word [p_h_buffers],0
00002673 EB12                           jmp     short sr7
                                    if18:
00002675 A1[1122]                       mov     ax,[p_buffers] ; we don't have any problem.
00002678 A3[9902]                       mov     [buffers],ax   ; now,let's set it really.

0000267B A1[1322]                       mov     ax,[p_h_buffers]
0000267E A3[9B02]                       mov     [h_buffers],ax

                                    ;   mov     al,[p_buffer_slash_x] ; M016
                                    ;   mov     [buffer_slash_x],al

00002681 A1[AF02]                       mov     ax,[linecount]
00002684 A3[B902]                       mov     [buffer_linenum],ax ; save the line number for the future use.
                                        ; 31/12/2022
                                        ;jmp    short sr7
                                        ; 03/01/2023
                                    sr7:
00002687 E9FCFE                         jmp     coff

                                    ;--------------------------------------------------------------------------
                                    ; break command
                                    ;--------------------------------------------------------------------------

                                    ;**************************************************************************
                                    ;                                                                         *
                                    ; function: parse the parameters of break = command.                     *
                                    ;                                                                         *
                                    ; input :                                                                 *
                                    ;   es:si -> parameters in command line.                                 *
                                    ; output:                                                                *
                                    ;   turn the control-c check 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_ctrl_break,on;                                      *
                                    ;          else                                  /*off         */ *
                                    ;             set p_ctrl_break,off;                                     *
                                    ;     else {show message;error_exit};                                    *
                                    ;   };                                                                   *
                                    ;   if (no error) then                                                  *
                                    ;      dos function call to set ctrl_break check according to          *
                                    ; };                                                                     *
                                    ;                                                                        *
                                    ;**************************************************************************

                                        ; 27/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                        ; (SYSINIT:22FFh)
                                    tryc:
0000268A 80FC43                         cmp     ah,CONFIG_BREAK ; 'C'
0000268D 7539                           jne     short trym

                                    ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:2527h)
                                    ; 27/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;%if 0
                                    ;ifdef      MULTI_CONFIG
0000268F E88A1F                         call    query_user              ; query the user if config_cmd
00002692 7234                           jc      short trym              ; has the CONFIG_OPTION_QUERY bit set
                                    ;endif
                                    ;%endif ; 27/10/2022

00002694 BF[1F22]                       mov     di,brk_parms
00002697 31C9                           xor     cx,cx
                                        ; 03/01/2023
                                        ;mov    dx,cx
                                    do22:
00002699 E8CB07                         call    sysinit_parse
0000269C 7303                           jnc     short if22              ; parse error
                                        ;call   badparm_p               ;  show message and end the search loop.
                                        ;;jmp   short sr22
                                        ; 31/12/2022
                                    ;sr22:
                                        ;jmp    coff
                                        ; 03/01/2023
0000269E E9B506                         jmp     badparm_p_coff
                                    if22:
000026A1 83F8FF                         cmp     ax,_$P_RC_EOL           ; end of line?
000026A4 7415                           je      short en22              ; then end the $endloop

                                        ;cmp    byte [result_val_itag],1
000026A6 803E[1822]01                   cmp     byte [result_val+_$P_Result_Blk.Item_Tag],1
000026AB 7507                           jne     short if26

000026AD C606[4422]01                   mov     byte [p_ctrl_break],1 ; turn it on
                                        ;jmp    short en26
```

```
                                    ; 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
```

```
34438                                   tryu_2:
34439 00002769 56                           push    si
34440 0000276A 06                           push    es
34441
34442                                       ; 08/09/2023 - Retro DOS 4.2 IO.SYS (Optimization)
34443                                       ; MSDOS 6.21 IO.SYS - SYSINIT:2623h
34444                                       ; PCDOS 7.1 IBMBIO.COM - SYSINIT:2B17h
34445                                   tryu_3:
34446 0000276B 268A04                       mov     al,[es:si]
34447 0000276E 3C0D                         cmp     al,cr
34448                                       ; 14/04/2024
34449                                       ;je     short tryu_4
34450                                       ; 08/09/2023 - Retro DOS 5.0 (PCDOS 7.1 IBMBIO.COM)
34451 00002770 740C                         je      short tryu_5
34452 00002772 3C0A                         cmp     al,lf
34453 00002774 740A                         je      short tryu_4
34454 00002776 E81120                       call    delim
34455 00002779 7405                         jz      short tryu_4
34456 0000277B 46                           inc     si
34457 0000277C EBED                         jmp     short tryu_3
34458
34459                                       ; 14/04/2024
34460                                       ; 08/09/2023 - Retro DOS 5.0 (PCDOS 7.1 IBMBIO.COM)
34461                                   tryu_5:
34462 0000277E B020                         mov     al,20h ; ' '   ; blank instead of cr
34463
34464                                   tryu_4:
34465                                       ; 11/12/2022
34466                                       ; ds = cs
34467 00002780 A2[6624]                     mov     [DevSavedDelim],al
34468                                       ;mov    [cs:DevSavedDelim],al ; Save the delimiter before replacing
34469                                                                     ;  it with null
34470                                       ; 18/12/2022
34471 00002783 29DB                         sub     bx,bx
34472 00002785 26881C                       mov     [es:si],bl ; 0
34473                                       ;mov    byte [es:si],0
34474
34475 00002788 07                           pop     es
34476 00002789 5E                           pop     si      ; 14/04/2024
34477
34478                                   ;--------------------------------------------------------------------------
34479                                   ; BEGIN PATCH TO CHECK FOR NON-EXISTANT UMBs   -- t-richj 7-21-92
34480                                   ;--------------------------------------------------------------------------
34481
34482                                   ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
34483                                   ; (SYSINIT:2642h)
34484                                   ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
34485                                   ;%if 0
34486                                   ; 10/07/2023 - Retro DOS v4.2 IO.SYS (optimized)
34487                                   ; MSDOS 6.21 IO.SYS - SYSINIT:2642h
34488                                   %if 1
34489                                       ; 01/01/2023
34490                                       ; ds = cs
34491 0000278A E8F00C                     call    UmbTest               ; See if UMBs are around...
34492                                       ; 01/01/2023
34493                                       ;jnc    short NrmTst          ; ...yep. So do that normal thang.
34494
34495                                       ;mov    byte [cs:DeviceHi],0  ; ...nope... so load low.
34496                                       ; 31/12/2022
34497                                       ; ds = cs, bx = 0
34498                                       ;mov    byte [DeviceHi],bl ; 0
34499                                       ;jmp    short LoadDevice
34500                                       ; 01/01/2023
34501 0000278D 7222                         jc      short LoadDevice ; bl = 0
34502                                   %endif
34503                                   ;%endif
34504                                   ;--------------------------------------------------------------------------
34505                                   ; END PATCH TO CHECK FOR NON-EXISTANT UMBs   -- t-richj 7-21-92
34506                                   ;--------------------------------------------------------------------------
34507
34508                                   NrmTst:
34509                                       ; 11/12/2022
34510                                       ; ds = cs
34511                                       ;;mov  byte [cs:DeviceHi],0
34512                                       ;mov    byte [DeviceHi],0
34513                                       ; 18/12/2022
34514                                       ; bx = 0
34515 0000278F 381E[4224]                   cmp     [DevUMB],bl ; 0
34516                                       ;cmp    byte [DevUMB],0
34517                                       ;;cmp   byte [cs:DevUMB],0    ; do we support UMBs
34518 00002793 741C                         je      short LoadDevice      ; no, we don't
34519                                       ;mov    byte [cs:DeviceHi],1
34520                                       ; 11/12/2022
34521                                       ;mov    byte [DeviceHi],1
34522                                       ; 18/12/2022
34523 00002795 FEC3                         inc     bl ; mov bl,1 ; (*)
34524                                       ; 11/12/2022
34525                                       ;jmp    short LoadDevice2     ; 11/12/2022
34526 00002797 EB18                         jmp     short LoadDevice
34527
34528                                   ;--------------------------------------------------------------------------
34529                                   ; device command
34530                                   ;--------------------------------------------------------------------------
34531
34532                                       ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
34533                                       ; (SYSINIT:2665h)
34534
34535                                       ; 28/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
34536                                       ; (SYSINIT:2401h)
34537                                   tryd:
34538                                       ; 11/12/2022
34539                                       ;xor    bx,bx ; 31/12/2022
34540                                       ;
34541 00002799 80FC44                          cmp     ah,CONFIG_DEVICE ; 'D'
34542 0000279C 7403                         je      short gotd
34543                                   skip_it2:
34544 0000279E E9FC02                       jmp     tryq
34545                                   gotd:
34546
34547                                   ; 31/12/2022 - Retro DOS v4.2
34548                                   ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
34549                                   ;%if 0
34550                                   ;ifdef      MULTI_CONFIG
34551 000027A1 E8781E                       call    query_user              ; query the user if config_cmd
34552 000027A4 72F8                         jc      short skip_it2        ; has the CONFIG_OPTION_QUERY bit set
34553                                   ;endif
34554                                   ;%endif ; 28/10/2022
34555
34556                                       ; 31/12/2022
34557 000027A6 29DB                         sub     bx,bx
34558                                       ; bx = 0
34559                                       ; 11/12/2022
34560                                       ; ds = cs
34561                                       ;mov    byte [DeviceHi],0
```

```
                                    ;mov    word [DevSizeOption],0
000027A8 891E[5224]                 mov     [DevSizeOption],bx ; 0
000027AC C606[6624]20               mov     byte [DevSavedDelim],' '
                                    ;mov    byte [cs:DeviceHi],0  ; not to be loaded in UMB ;M007
                                    ;mov    word [cs:DevSizeOption],0
                                    ;mov    byte [cs:DevSavedDelim],' ' ; In case of DEVICE= the null has to
                                                                        ;  be replaced with a ' '
                                LoadDevice:                                 ; device= or devicehigh= command.
                                    ; 11/12/2022
                                    ;mov    byte [DeviceHi],0
000027B1 881E[5124]                 mov     byte [DeviceHi],bl    ; 0 or 1 (*)
                                LoadDevice2:
                                    ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;
                                    ;push    cs
                                       ;pop     ds
                                    ;
                                    ;mov    [bpb_addr],si         ; pass the command line to the dvice
                                    ;mov    [bpb_addr+2],es
                                    ;
                                    ;mov    [DevCmdLine],si              ; save it for ourself
                                    ;mov    [DevCmdLine+2],es
                                    ;
                                    ;mov    byte [driver_units],0 ; clear total block units for driver

                                    ; 11/12/2022
                                    ; ds = cs
                                    ;mov    bx,cs
                                    ;mov    ds,bx

                                    ;mov    [cs:bpb_addr],si      ; pass the command line to the dvice
000027B5 8936[8103]                 mov     [bpb_addr],si
                                    ;mov    [cs:bpb_addr+2],es
000027B9 8C06[8303]                 mov     [bpb_addr+2],es

                                    ;mov    [cs:DevCmdLine],si    ; save it for ourself
000027BD 8936[6224]                 mov     [DevCmdLine],si
                                    ;mov    [cs:DevCmdLine+2],es
000027C1 8C06[6424]                 mov     [DevCmdLine+2],es

                                    ; 31/12/2022 - Retro DOS v4.2
000027C5 C606[6A19]00               mov     byte [driver_units],0 ; clear total block units for driver

000027CA E82520                     call    round

000027CD E8910E                     call    SizeDevice
000027D0 723F                       jc      short BadFile

                                    ; 11/12/2022
                                    ; ds = cs

                                ; - Begin DeviceHigh primary logic changes ---------------------------------

                                ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:26A4h)

                                ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
000027D2 C606[4124]01               mov     byte [ConvLoad],1     ; Doesn't matter if DeviceHi==0

                                    ; 22/07/2023
                                    ;mov    al,[DeviceHi]         ; If not using upper memory,
000027D7 800E[5124]00               or      byte [DeviceHi],0     ; Skip all this and go on to
                                    ; 10/07/2023
                                    ;or     al,al
000027DC 741E                       jz      short DevConvLoad     ; the actual load.

                                    ;call   GetLoadUMB            ; Returns first UMB spec'ed in AX
000027DE A0[FF23]                   mov     al,[UmbLoad]   ; 19/04/2019 - Retro DOS v4.0

000027E1 3CFF                       cmp     al,-1                 ; If umb0 not specified, it's old style
000027E3 7417                       jz      short DevConvLoad     ; so load high even if SIZE= is smaller

000027E5 FE0E[4124]                 dec     byte [ConvLoad] ; 0   ; They specified /L, so use new loader

000027E9 E8590A                     call    GetLoadSize           ; Returns size of first UMB specified
000027EC 09C0                       or      ax,ax
000027EE 7406                       jz      short tryd_1          ; If size1 not specified, nada to do:

000027F0 3B06[3324]                 cmp     ax,[DevSize]          ; /L:...,Size < DevSize?
000027F4 7D06                       jge     short DevConvLoad
                                tryd_1:
000027F6 A1[3324]                   mov     ax,[DevSize]          ; Size < DevSize, so write DevSize as
000027F9 E8550A                     call    StoLoadSize           ; minsize for load UMB.

                                ;%endif ; 28/10/2022

                                ; - End DeviceHigh primary logic changes -----------------------------------

                                DevConvLoad:
                                    ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
000027FC E8660D                     call    InitDevLoad

                                    ; 11/12/2022
                                    ; ds = cs
000027FF A1[3524]                   mov     ax,[DevLoadAddr]
00002802 0306[3324]                 add     ax,[DevSize]
00002806 7206                       jc      short NoMem
00002808 3906[3724]                 cmp     [DevLoadEnd],ax
0000280C 7315                       jae     short LoadDev

                                    ; 11/12/2022
                                    ;mov    ax,[cs:DevLoadAddr]
                                    ;add    ax,[cs:DevSize]
                                    ;jc     short NoMem
                                    ;cmp    [cs:DevLoadEnd],ax
                                    ;jae    short LoadDev
                                NoMem:
                                    ; 11/12/2022
                                    ; ds = cs
                                    ;jmp    mem_err
0000280E E92020                     jmp     mem_err2

                                BadFile:
                                    ; 28/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;;call  RetFromUM             ; Does nothing if didn't call HideUMBs
                                    ;;cmp   byte [es:si],' '
                                       ;;jae      short tryd_2
                                    ; 31/12/2022
                                    ;cmp    byte [es:si],0Dh      ; cr
                                       ;jne       short tryd_2
                                    ;jmp    badop
                                    ; 31/12/2022
                                    ; ds = cs
```

```
                                    ; 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
```

```
34810                                   ;lds    bx,[cs:DOSINFO]             ; ds:bx -> sys_var
34811                                   ; 31/12/2022
34812                                   ; ds = cs
34813                                   ; 08/09/2023
34814                                   ;lds    bx,[DOSINFO]          ; ds:bx -> sys_var
34815
34816                                   ;mov    al,[cs:drivenumber]   ; temporarily use this next drv value
34817                                   ;mov    [cs:devdrivenum],al   ; pass drive number in packet to driver
34818                                   ;mov    ah,al
34819
34820                                   ; 08/09/2023
34821                                   ; ds = cs
34822 00002874 A0[8503]                mov     al,[drivenumber]      ; temporarily use this next drv value
34823 00002877 A2[8503]                mov     [devdrivenum],al      ; pass drive number in packet to driver
34824 0000287A 88C4                    mov     ah,al
34825 0000287C C51E[6D02]              lds     bx,[DOSINFO]          ; ds:bx -> sys_var
34826
34827 00002880 874720                  xchg    ax,[bx+SYSI_NUMIO]    ; swap with existing values
34828                                   ; 31/12/2022
34829                                   ;pop    ds
34830
34831 00002883 50                      push    ax                    ; save real sysi_numio/ncds in ax
34832
34833                                 ;%endif ; 29/10/2022
34834
34835                                   ; 29/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
34836                                   ; (SYSINIT:24B9h)
34837
34838 00002884 BB0600                  mov     bx,SYSDEV.STRAT ; 6
34839 00002887 E8B01F                  call    calldev               ; calldev (sdevstrat);
34840 0000288A BB0800                  mov     bx,SYSDEV.INT ; 8
34841 0000288D E8AA1F                  call    calldev               ; calldev (sdevint);
34842
34843                                   ; 11/12/2022
34844                                   ; ds <> cs (from calldev) ; 31/12/2022
34845
34846                                 ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
34847                                 ; (SYSINIT:2773h)
34848                                 ; 29/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
34849                                 ;%if 0
34850 00002890 58                        pop     ax                    ; get real sysi_numio value
34851                                     ; 31/12/2022
34852                                     ;push   ds
34853 00002891 2EC51E[6D02]              lds     bx,[cs:DOSINFO]             ; ds:bx -> sys_var
34854 00002896 894720                    mov     [bx+SYSI_NUMIO],ax    ; swap with existing values
34855                                     ; 31/12/2022
34856                                     ;pop    ds
34857
34858                                 ;%endif ; 29/10/2022
34859
34860                                     ; 11/12/2022
34861 00002899 0E                        push    cs
34862 0000289A 1F                        pop     ds
34863
34864 0000289B E89C06                    call    TrueInt12Mem
34865
34866                                     ; 11/12/2022
34867                                     ; ds = cs
34868                                     ;mov    ax,[cs:break_addr]    ; move break addr from the req packet
34869                                     ;mov    [cs:DevBrkAddr],ax
34870                                     ;mov    ax,[cs:break_addr+2]
34871                                     ;mov    [cs:DevBrkAddr+2],ax
34872 0000289E A1[7D03]                  mov     ax,[break_addr]
34873 000028A1 A3[3D24]                  mov     [DevBrkAddr],ax
34874 000028A4 A1[7F03]                  mov     ax,[break_addr+2]
34875 000028A7 A3[3F24]                  mov     [DevBrkAddr+2],ax
34876
34877                                     ; 29/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
34878                                     ;call   RetFromUM             ; There we go... all done.
34879                                     ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
34880                                     ; (SYSINIT:2791h)
34881 000028AA E8150E                    call    RetFromUM             ; There we go... all done.
34882
34883                                     ; 31/12/2022
34884                                     ; ds = cs
34885
34886                                     ; 11/12/2022
34887 000028AD 803E[4224]00              cmp     byte [DevUMB],0
34888                                     ;cmp    byte [cs:DevUMB],0
34889 000028B2 7403                      je      short tryd_3
34890 000028B4 E80010                    call    AllocUMB
34891                                     ; 31/12/2022
34892                                     ; ds = cs
34893                                 tryd_3:
34894
34895                                 ;ifndef ROMDOS
34896                                 ;------ If we are waiting to be moved into hma lets try it now !!!
34897                                     ; 11/12/2022
34898                                     ; ds = cs
34899
34900
34901                                     ;cmp    byte [cs:runhigh],0FFh
34902 000028B7 803E[6C02]FF              cmp     byte [runhigh],0FFh ; 11/12/2022
34903 000028BC 7503                      jne     short tryd_4
34904
34905                                     ; 11/12/2022
34906                                     ; ds = cs
34907 000028BE E8D7E1                    call    TryToMovDOSHi         ; move DOS into HMA if reqd
34908                                 tryd_4:
34909                                 ;endif ; ROMDOS
34910
34911 000028C1 5E                        pop     si
34912 000028C2 1F                        pop     ds
34913 000028C3 C60400                    mov     byte [si],0           ; *p = 0;
34914
34915 000028C6 0E                        push    cs
34916 000028C7 1F                        pop     ds
34917
34918 000028C8 EB1F                      jmp     short was_device_com
34919
34920                                 ;--------------------------------------------------------------------------
34921
34922                                 ; 02/04/2019 - Retro DOS v4.0
34923
34924                                 ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
34925                                 ; (SYSINIT:27B3h)
34926
34927                                 badnumblock:
34928 000028CA 0E                        push    cs
34929 000028CB 1F                        pop     ds
34930                                 badnumblock2:       ; 31/12/2022 (ds=cs)
34931 000028CC BA[7051]                  mov     dx,badblock
34932 000028CF E88221                    call    print
34933
```

```
                                ;------ fall thru ---------------------------------------------------------------
                                
                                    ; 31/12/2022 - Retro DOS v4.2
                                
                                erase_dev_do:                           ; modified to show message "error in config.sys..."
                                
                                    ;call  CheckDoubleSpace ; MSDOS 6.21 IO.SYS SYSINIT:27BBh
                                                            ; (Note: 'call CheckDoubleSpace'
                                                            ; has been removed at 'erase_dev_do:' pos
                                                            ; in PCDOS 7.1 IBMBIO.COM - SYSINIT:2CBAh)
                                                            ; Erdogan Tan - 10/07/2023
000028D2 5E                         pop     si ; ++
000028D3 07                         pop     es ; + ; 31/12/2022
                                
000028D4 0E                         push    cs
000028D5 1F                         pop     ds
                                
                                skip1_resetmemhi:
                                    ; 11/12/2022
                                    ; ds = cs
000028D6 833E[8603]00               cmp     word [configmsgflag],0
                                    ;cmp    word [cs:configmsgflag],0
000028DB 7409                       je      short no_error_line_msg
                                
000028DD E8DA05                     call    error_line              ; no "error in config.sys" msg for device driver. dcr d493
                                    ; 11/12/2022
                                    ; ds = cs
                                    ;mov    word [cs:configmsgflag],0
000028E0 C706[8603]0000             mov     word [configmsgflag],0; set the default value again.
                                
                                no_error_line_msg:
000028E6 E99DFC                     jmp     coff
                                
                                ;--------------------------------------------------------------------------------
                                
                                was_device_com:
                                    ; 14/12/2022
                                    ; ds = cs
000028E9 A1[3F24]                   mov     ax,[DevBrkAddr+2]
                                    ;mov    ax,[cs:DevBrkAddr+2] ; 13/05/2019
000028EC 3B06[3724]                 cmp     ax,[DevLoadEnd]
                                    ;cmp    ax,[cs:DevLoadEnd]
000028F0 7605                       jbe     short breakok
                                
000028F2 5E                         pop     si
000028F3 07                         pop     es
000028F4 E91AFF                     jmp     BadFile
                                
                                breakok:
                                    ; 14/12/2022
                                    ; ds = cs
000028F7 C43E[6D02]                 les     di,[DOSINFO]
000028FB C516[3924]                 lds     dx,[DevEntry]
                                    ;lds    dx,[cs:DevEntry]        ;set ds:dx to header
000028FF 89D6                       mov     si,dx
                                
                                    ; 14/11/2022
                                    ;les    di,[cs:DOSINFO]         ;es:di point to dos info
                                
                                    ; 14/12/2022
                                    ; ds <> cs
                                
                                    ;mov    ax,[si+4]
00002901 8B4404                     mov     ax,[si+SYSDEV.ATT]      ;get attributes
                                    ; 12/12/2022
00002904 F6C480                     test    ah,DEVTYP>>8 ; 80h
                                    ;test   ax,DEVTYP ; 8000h       ;test if block dev
00002907 7426                       jz      short isblock
                                
                                ;------ lets deal with character devices
                                
00002909 2E800E[6919]02             or      byte [cs:setdevmarkflag],for_devmark ; 2
0000290F E8F40D                     call    DevSetBreak             ;go ahead and alloc mem for device
                                jc_edd:
00002912 72BE                       jc      short erase_dev_do      ;device driver's init routine failed.
                                
                                    ; 12/12/2022
00002914 A801                       test    al,ISCIN
                                    ;test   ax,ISCIN ; 1            ;is it a console in?
00002916 7408                       jz      short tryclk
                                
00002918 2689550C                   mov     [es:di+SYSI_CON],dx   ; es:di+12
0000291C 268C5D0E                   mov     [es:di+SYSI_CON+2],ds ; es:di+14
                                tryclk:
                                    ; 12/12/2022
00002920 A808                       test    al,ISCLOCK
                                    ;test   ax,ISCLOCK ; 8          ;is it a clock device?
00002922 7408                       jz      short golink
                                
00002924 26895508                   mov     [es:di+SYSI_CLOCK],dx ; es:di+8
00002928 268C5D0A                   mov     [es:di+SYSI_CLOCK+2],ds ; es:di+10
                                golink:
0000292C E9DF00                     jmp     linkit
                                
                                ;------ deal with block device drivers
                                
                                isblock:
0000292F 2EA0[7C03]                 mov     al,[cs:unitcount]       ;if no units found,erase the device
00002933 08C0                       or      al,al
00002935 749B                       jz      short erase_dev_do
                                    ;mov    [si+10],al
00002937 88440A                     mov     [si+SYSDEV.NAME],al   ; number of units in name field
                                    ; 29/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;add    [cs:driver_units],al
                                    ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
0000293A 2E0006[6A19]               add     [cs:driver_units],al  ; keep total for all drivers in file
                                perdrv:
0000293F 98                         cbw                             ; warning no device > 127 units
00002940 89C1                       mov     cx,ax
00002942 88E6                       mov     dh,ah
                                    ;mov    dl,[es:di+32]
00002944 268A5520                   mov     dl,[es:di+SYSI_NUMIO] ;get number of devices
00002948 88D4                       mov     ah,dl
0000294A 00C4                       add     ah,al                   ; check for too many devices
0000294C 80FC1A                     cmp     ah,26                   ; 'A' - 'Z' is 26 devices
0000294F 7603                       jbe     short ok_block
00002951 E976FF                     jmp     badnumblock
                                
                                ok_block:
00002954 2E800E[6919]02             or      byte [cs:setdevmarkflag],for_devmark ; 2
0000295A E8A90D                     call    DevSetBreak             ; alloc the device
0000295D 72B3                       jc      short jc_edd
0000295F 26004520                   add     [es:di+SYSI_NUMIO],al ; update the amount
                                
```

```
35058 00002963 2E0006[8503]               add     [cs:drivenumber],al   ; remember amount for next device
35059 00002968 2EC51E[8103]               lds     bx,[cs:bpb_addr]      ; point to bpb array
35060                                   perunit:
35061 0000296D 2EC42E[6D02]               les     bp,[cs:DOSINFO]
35062                                     ;les    bp,[es:bp+SYSI_DPB]   ; get first dpb
35063                                     ; 11/12/2022
35064 00002972 26C46E00                   les     bp,[es:bp]
35065                                     ; 29/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
35066                                     ;les    bp,[es:bp+0]          ; [es:bp+SYSI_DPB]
35067                                   scandpb:
35068                                     ;cmp    word [es:bp+25],-1
35069 00002976 26837E19FF                 cmp     word [es:bp+DPB.NEXT_DPB],-1
35070 0000297B 7406                       je      short foundpb
35071                                     ;les    bp,[es:bp+25]
35072 0000297D 26C46E19                   les     bp,[es:bp+DPB.NEXT_DPB]
35073 00002981 EBF3                       jmp     short scandpb
35074                                   foundpb:
35075 00002983 2EA1[3D24]                 mov     ax,[cs:DevBrkAddr]
35076 00002987 26894619                   mov     [es:bp+DPB.NEXT_DPB],ax
35077 0000298B 2EA1[3F24]                 mov     ax,[cs:DevBrkAddr+2]
35078 0000298F 2689461B                   mov     [es:bp+DPB.NEXT_DPB+2],ax
35079
35080 00002993 2EC42E[3D24]               les     bp,[cs:DevBrkAddr]
35081 00002998 2E8306[3D24]3D             add     word [cs:DevBrkAddr],DPBSIZ ; 33
35082                                                                 ; 08/09/2023
35083                                                                 ; (61 in PCDOS 7.1 IBMBIO.COM)
35084 0000299E E8440D                     call    RoundBreakAddr
35085
35086 000029A1 26C74619FFFF               mov     word [es:bp+DPB.NEXT_DPB],-1
35087 000029A7 26C64618FF                 mov     byte [es:bp+DPB.FIRST_ACCESS],-1
35088
35089 000029AC 8B37                       mov     si,[bx]               ;ds:si points to bpb
35090 000029AE 43                         inc     bx
35091 000029AF 43                         inc     bx                    ;point to next guy
35092                                     ;mov    [es:bp+DPB.DRIVE],dx
35093                                     ; 11/12/2022
35094 000029B0 26895600                   mov     [es:bp],dx ; 13/05/2019
35095                                     ; 29/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
35096                                     ;mov    [es:bp+0],dx          ; [es:bp+DPB.DRIVE]
35097
35098                                     ; 13/04/2024 - Retro DOS v5.0
35099                                     ; PCDOS 7.1 IBMBIO.COM
35100                                     ;;;
35101 000029B4 52                         push    dx
35102 000029B5 51                         push    cx              ; initialize FAT32 extended DPB parameters/fields
35103 000029B6 BA5241                     mov     dx,4152h        ; 'AR' signature for FAT32 extended DPB
35104 000029B9 31C9                       xor     cx,cx   ; 0
35105                                     ;mov    [es:bp+1Dh],cx
35106 000029BB 26894E1D                   mov     [es:bp+DPB.NEXT_FREE],cx ; last allocated cluster #
35107                                     ;cmp    [si+0Bh],cx     ; BPB.fatsecs16
35108 000029BF 394C0B                     cmp     [si+A_BPB.SECTORSPERFAT],cx ; 0
35109 000029C2 7514                       jnz     short set_dpb   ; FAT DPB (33 bytes)   -jnz-
35110                                                             ; FAT32 DPB (61 bytes) -jz-
35111                                     ;mov    [es:bp+39h],cx
35112 000029C4 26894E39                   mov     [es:bp+DPB.FAT32_NXTFREE],cx ; 0
35113                                     ;mov    [es:bp+3Bh],cx
35114 000029C8 26894E3B                   mov     [es:bp+DPB.FAT32_NXTFREE+2],cx ; 0
35115 000029CC 49                         dec     cx      ; 0FFFFh ; -1
35116                                     ;mov    [es:bp+1Fh],cx
35117 000029CD 26894E1F                   mov     [es:bp+DPB.FREE_CNT],cx ; -1 = unknown
35118                                     ;mov    [es:bp+21h],cx
35119 000029D1 26894E21                   mov     [es:bp+DPB.FREE_CNT+2],cx ; -1 = unknown
35120 000029D5 B95845                     mov     cx,4558h        ; 'EX' signature for FAT32 extended DPB
35121                                   set_dpb:
35122                                     ;;;
35123
35124 000029D8 B453                       mov     ah,SETDPB ; 53h               ;hidden system call
35125 000029DA CD21                       int     21h
35126                                             ; DOS - 2+ internal - TRANSLATE BIOS PARAMETER BLOCK
35127                                             ; DS:SI -> BPB (BIOS Parameter Block)
35128                                             ; ES:BP -> buffer for DOS Drive Parameter Block
35129                                     ; 13/04/2024
35130                                     ;;;
35131 000029DC 59                         pop     cx
35132 000029DD 5A                         pop     dx
35133                                     ;;;
35134
35135                                     ;mov    ax,[es:bp+2]
35136 000029DE 268B4602                   mov     ax,[es:bp+DPB.SECTOR_SIZE]
35137 000029E2 06                         push    es
35138 000029E3 2EC43E[6D02]               les     di,[cs:DOSINFO]       ;es:di point to dos info
35139                                     ;cmp    ax,[es:di+10h]
35140 000029E8 263B4510                   cmp     ax,[es:di+SYSI_MAXSEC]
35141 000029EC 07                         pop     es
35142                                     ; 13/04/2024
35143                                     ;jna    short iblk_1
35144                                     ;jmp    bad_bpb_size_sector
35145                                     ; 29/10/2022
35146 000029ED 777F                       ja      short bad_bpb_size_sector
35147                                   iblk_1:
35148 000029EF 1E                         push    ds
35149 000029F0 52                         push    dx
35150
35151 000029F1 2EC516[3924]               lds     dx,[cs:DevEntry]
35152                                     ;mov    [es:bp+13h],dx
35153 000029F6 26895613                   mov     [es:bp+DPB.DRIVER_ADDR],dx
35154                                     ;mov    [es:bp+15h],ds
35155 000029FA 268C5E15                   mov     [es:bp+DPB.DRIVER_ADDR+2],ds
35156
35157 000029FE 5A                         pop     dx
35158 000029FF 1F                         pop     ds
35159
35160 00002A00 42                         inc     dx
35161 00002A01 FEC6                       inc     dh
35162                                     ;loop   perunit
35163                                     ; 13/04/2024
35164                                     ;;;
35165 00002A03 49                         dec     cx              ; cx = cx - 1
35166                                                             ; cx = remain count from [cs:unitcount]
35167 00002A04 7403                       jz      short iblk_2    ; cx = 0 -> done
35168 00002A06 E964FF                     jmp     perunit         ; loop until cx is 0
35169                                   iblk_2:
35170                                     ;;;
35171
35172 00002A09 0E                         push    cs
35173 00002A0A 1F                         pop     ds
35174
35175 00002A0B E895E3                     call    TempCDS                 ; set cds for new drives
35176                                     ; 31/12/2022
35177                                     ; ds <> cs
35178                                   linkit:
35179 00002A0E 2EC43E[6D02]               les     di,[cs:DOSINFO]       ;es:di = dos table
35180 00002A13 268B4D22                   mov     cx,[es:di+SYSI_DEV]   ;dx:cx = head of list
35181 00002A17 268B5524                   mov     dx,[es:di+SYSI_DEV+2]
```

```
35182
35183 00002A1B 2EC536[3924]               lds     si,[cs:DevEntry]      ;ds:si = device location
35184 00002A20 26897522                   mov     [es:di+SYSI_DEV],si   ;set head of list in dos
35185 00002A24 268C5D24                   mov     [es:di+SYSI_DEV+2],ds
35186 00002A28 8B04                       mov     ax,[si]               ;get pointer to next device
35187 00002A2A 2EA3[3924]                 mov     [cs:DevEntry],ax      ;and save it
35188
35189 00002A2E 890C                       mov     [si],cx               ;link in the driver
35190 00002A30 895402                     mov     [si+2],dx
35191                                   enddev:
35192 00002A33 5E                         pop     si
35193 00002A34 07                         pop     es
35194 00002A35 40                         inc     ax                    ;ax = ffff (no more devs if yes)?
35195 00002A36 740B                       jz      short coffj3
35196
35197 00002A38 2EFE06[6619]               inc     byte [cs:multdeviceflag] ; possibly multiple device driver.
35198 00002A3D E8E80C                     call    DevBreak              ; M009
35199                                     ; 11/12/2022
35200                                     ; ds = cs (DevBreak)
35201
35202                                     ; 03/04/2019 - Retro DOS v4.0
35203                                     ; MSDOS 6.21 IO.SYS - SYSINIT:290Dh
35204 00002A40 E9EDFD                     jmp     goodld                ; otherwise pretend we loaded it in
35205                                   coffj3:
35206                                     ; 18/12/2022
35207                                     ; ax = 0
35208 00002A43 2EA2[6619]                 mov     [cs:multdeviceflag],al ; 0
35209                                     ;mov    byte [cs:multdeviceflag],0 ; reset the flag
35210 00002A47 E8DE0C                     call    DevBreak
35211                                     ; 11/12/2022
35212                                     ; ds = cs (DevBreak)
35213
35214                                     ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
35215                                     ; (SYSINIT:2919h)
35216                                     ; 11/07/2023
35217 00002A4A E80204                     call    CheckProtmanArena
35218
35219                                     ; 02/11/2022 (MSDOS 5.0 IO.SYS compatibility)
35220                                     ;;call  CheckProtmanArena     ; adjust alloclim if Protman$ just
35221                                   ;                               ;  created a bogus arena to try
35222                                   ;                               ;  to protect some of its resident-
35223                                   ;                               ;  init code.
35224                                     ; 13/04/2024 - Retro DOS v5.0
35225                                     ; PCDOS 7.1 IBMBIO.COM
35226                                     ;;call  CheckDoubleSpace
35227                                     ;jmp    coff
35228
35229                                   ;--------------------------------------------------------------------------
35230
35231                                     ; 13/04/2024 - Retro DOS v5.0
35232                                     ; PCDOS 7.1 IBMBIO.COM
35233                                     ;;;
35234
35235                                   CheckDoubleSpace:
35236
35237                                   ;;  ifdef   dblspace_hooks
35238
35239                                   ;   Now check for two special MagicDrv cases:
35240                                   ;
35241                                   ;       a) the last driver load was MagicDrv final placement:
35242                                   ;      -> add number of MagicDrv reserved drives to drivenumber
35243                                   ;
35244                                   ;       b) MagicDrv is currently in temporary home:
35245                                   ;          -> call it to give it a chance to mount and shuffle drives
35246
35247                                     ;cmp    byte [cs:MagicHomeFlag],0 ; already home?
35248                                     ; 13/04/2024 (PCDOS 7.1 IBMBIO.COM)
35249 00002A4D 2EF606[6724]01             test    byte [cs:MagicHomeFlag],1 ; already home?
35250 00002A53 7545                       jnz     short no_more_magic_calls ;  nothing more to do if so
35251
35252                                   ;   Now inquire of driver whether it is present, and final located
35253
35254                                     ;mov    ax,multMagicdrv ; 4A11h
35255                                     ;mov    bx,MD_VERSION ; 0
35256                                     ;int    2fh                   ; ch = number of MagicDrv drive letters
35257                                     ;or     ax,ax                 ; is it there?
35258                                     ; 13/04/2024 (PCDOS 7.1 IBMBIO.COM)
35259                                     ;;;
35260 00002A55 E8FEEE                     call    get_dblspace_version ; is it there?
35261                                     ;jnz    short no_more_magic_calls ; done if not
35262 00002A58 750B                       jnz     short set_magichomeflag
35263                                     ;;;
35264
35265 00002A5A F7C20080                   test    dx,8000h              ; is it final placed?
35266 00002A5E 751C                       jnz     short magic_not_yet_home ;  skip if not
35267
35268                                   ;   Okay, now the driver is final placed!  Set the flag so we
35269                                   ;   don't keep checking it, and add its number of drive letters
35270                                   ;   to drivenumber.
35271
35272                                     ; 13/04/2024 (PCDOS 7.1 IBMBIO.COM)
35273                                     ;mov    byte [cs:MagicHomeFlag],0ffh ; set the flag!
35274 00002A60 2E002E[8503]               add     [cs:drivenumber],ch   ; add number of MagicDrv volumes to
35275                                                                   ;  the drive number we'll pass to the
35276                                                                   ;  next loadable block device.
35277                                     ;jmp    short no_more_magic_calls ; and finished.
35278
35279                                     ;;;
35280                                   set_magichomeflag:
35281 00002A65 2EC606[6724]01             mov     byte [cs:MagicHomeFlag],1 ; set the flag!
35282 00002A6B E918FB                     jmp     coff
35283                                     ;;;
35284
35285                                   ; 03/04/2019 - Retro DOS v4.0
35286
35287                                   bad_bpb_size_sector:
35288 00002A6E 5E                         pop     si
35289 00002A6F 07                         pop     es
35290 00002A70 BA[9250]                   mov     dx,badsiz_pre
35291 00002A73 BB[7050]                   mov     bx,crlfm
35292 00002A76 E8B91F                     call    prnerr
35293
35294 00002A79 E90AFB                     jmp     coff
35295
35296                                   magic_not_yet_home:
35297 00002A7C 06                         push    es
35298 00002A7D 56                         push    si
35299
35300 00002A7E 2E8B0E[6403]               mov     cx,[cs:memhi]         ; pass it a work buffer
35301 00002A83 2E8B16[A502]               mov     dx,[cs:ALLOCLIM]      ;   address in cx (segment)
35302 00002A88 29CA                       sub     dx,cx                 ;   for len dx (paragraphs)
35303
35304 00002A8A BB0200                     mov     bx,2
35305 00002A8D 2EA0[6A19]                 mov     al,[cs:driver_units]  ; shuffle magicdrives and new drives
```

```
                                                        ;   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.
```

```
35430                                   ; 14/12/2022
35431 00002AEB C536[1B22]               lds     si,[rv_dword]
35432 00002AEF BF[DD4A]                 mov     di,cntry_drv
35433 00002AF2 E82C1F                   call    move_asciiz
35434
35435 00002AF5 5F                       pop     di
35436 00002AF6 5E                       pop     si
35437 00002AF7 07                       pop     es
35438 00002AF8 1F                       pop     ds
35439                                 en56:
35440 00002AF9 EBBC                     jmp     short do52
35441                                 sr52:
35442                                   ; 14/12/2022
35443                                   ; ds = cs
35444 00002AFB 833E[7A22]FF             cmp     word [p_cntry_code],-1
35445                                   ;cmp    word [cs:p_cntry_code],-1     ; had a parse error?
35446 00002B00 7509                     jne     short tryq_open
35447 00002B02 E981FA                   jmp     coff
35448
35449                                 tryqbad:                          ;"invalid country code or code page"
35450 00002B05 F9                       stc
35451 00002B06 BA[D950]                 mov      dx,badcountry
35452 00002B09 EB79                     jmp      tryqchkerr
35453
35454                                 tryq_open:
35455                                   ; 14/12/2022
35456                                   ; ds = cs
35457 00002B0B 803E[DD4A]00             cmp     byte [cntry_drv],0
35458                                   ;cmp    byte [cs:cntry_drv],0
35459 00002B10 7405                     je      short tryq_def
35460 00002B12 BA[DD4A]                 mov     dx,cntry_drv
35461 00002B15 EB03                     jmp     short tryq_openit
35462
35463                                 tryq_def:
35464 00002B17 BA[DF4A]                 mov     dx,cntry_root
35465                                 tryq_openit:
35466 00002B1A B8003D                   mov     ax,3D00h                  ;open a file
35467 00002B1D F9                       stc
35468 00002B1E CD21                     int     21h
35469 00002B20 7242                     jc      short tryqfilebad         ;open failure
35470
35471                                   ; 14/12/2022
35472                                   ; ds = cs
35473 00002B22 A3[5C03]                 mov     [cntryfilehandle],ax
35474                                   ;mov    [cs:cntryfilehandle],ax       ;save file handle
35475 00002B25 89C3                     mov     bx,ax
35476 00002B27 A1[7A22]                 mov     ax,[p_cntry_code]
35477 00002B2A 8B16[7C22]               mov     dx,[p_code_page]
35478                                   ;mov    ax,[cs:p_cntry_code]
35479                                   ;mov    dx,[cs:p_code_page]   ;now,ax=country id,bx=filehandle
35480                                   ;mov    cx,[cs:memhi]
35481 00002B2E 8B0E[6403]               mov     cx,[memhi]
35482 00002B32 81C18001                 add     cx,384                    ;need 6k buffer to handle country.sys
35483                                                                     ;M023
35484                                   ; 14/12/2022
35485                                   ; ds = cs
35486 00002B36 3B0E[A502]               cmp     cx,[ALLOCLIM]
35487                                   ;cmp    cx,[cs:ALLOCLIM]
35488 00002B3A 7745                     ja      short tryqmemory          ;cannot allocate the buffer for country.sys
35489
35490 00002B3C BE[DD4A]                 mov     si,cntry_drv              ;ds:si -> cntry_drv
35491 00002B3F 803C00                   cmp     byte [si],0               ;default path?
35492 00002B42 7502                     jne     short tryq_set_for_dos
35493
35494 00002B44 46                        inc     si
35495 00002B45 46                       inc     si                        ;ds:si -> cntry_root
35496
35497                                 tryq_set_for_dos:
35498                                   ; 14/12/2022
35499                                   ; ds = cs
35500 00002B46 C43E[7902]               les     di,[sysi_country]
35501                                   ;les    di,[cs:sysi_country]  ;es:di -> country info tab in dos
35502 00002B4A 57                       push    di                        ;save di
35503                                   ;add    di,8
35504 00002B4B 83C708                   add     di,country_cdpg_info.ccPath_CountrySys ; 8
35505 00002B4E E8D01E                   call    move_asciiz               ;set the path to country.sys in dos.
35506 00002B51 5F                       pop     di                        ;es:di -> country info tab again.
35507
35508                                   ; 14/12/2022
35509 00002B52 8B0E[6403]               mov     cx,[memhi]
35510                                   ;mov    cx,[cs:memhi]
35511 00002B56 8ED9                     mov     ds,cx
35512 00002B58 31F6                     xor     si,si                     ;ds:si -> 2k buffer to be used.
35513 00002B5A E8601D                   call    setdoscountryinfo         ;now do the job!!!
35514                                   ; ds <> cs ; 14/12/2022
35515 00002B5D 7325                     jnc     short tryqchkerr          ;read error or could not find country,code page combination
35516
35517 00002B5F 83F9FF                   cmp     cx,-1                     ;could not find matching country_id,code page?
35518 00002B62 74A1                     je      short tryqbad             ;then "invalid country code or code page"
35519
35520                                 tryqfilebad:
35521 00002B64 0E                       push    cs
35522 00002B65 07                       pop     es
35523 00002B66 2E803E[DD4A]00           cmp     byte [cs:cntry_drv],0 ;is the default file used?
35524 00002B6C 7405                     je      short tryqdefbad
35525
35526 00002B6E BE[DD4A]                 mov     si,cntry_drv
35527 00002B71 EB03                     jmp     short tryqbadload
35528
35529                                 tryqdefbad:                       ;default file has been used.
35530 00002B73 BE[DF4A]                 mov     si,cntry_root             ;es:si -> \country.sys in sysinit_seg
35531                                 tryqbadload:
35532 00002B76 E8B31E                   call    badload                   ;ds will be restored to sysinit_seg
35533                                   ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
35534                                   ; (SYSINIT:2A69h)
35535 00002B79 8B0E[A302]               mov     cx,[CONFBOT] ; ds = cs (from badload)
35536                                   ;mov    cx,[cs:CONFBOT]
35537                                   ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
35538                                   ;mov    cx,[cs:top_of_cdss]
35539                                   ; 11/12/2022
35540                                   ; ds = cs
35541                                   ;mov    cx,[top_of_cdss]  ; mov cx,[CONFBOT]
35542 00002B7D 8EC1                     mov     es,cx                     ;restore es -> confbot.
35543 00002B7F EB13                     jmp     short coffj4
35544
35545                                 tryqmemory:
35546 00002B81 BA[1C51]                 mov     dx,insufmemory
35547                                 tryqchkerr:
35548                                   ;mov    cx,[cs:CONFBOT]
35549                                   ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
35550                                   ;mov    cx,[cs:top_of_cdss]
35551                                   ; 12/12/2022
35552 00002B84 0E                       push    cs
35553 00002B85 1F                       pop     ds
```

```
                                    ; 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.
                        ;   any combinations that are not within the specified limits will
                        ;   result in "unrecognized command" error.
                        ;-----------------------------------------------------------------------

                        ;************************************************************************
                        ;                                                                       *
                        ; function: parse the parameters of stacks= command.                   *
                        ;       the minimum value for "number of stacks" and "stack size" is  *
                        ;       8 and 32 each.  in the definition of sysparse value list,they *
                        ;       are set to 0.  this is for accepting the exceptional case of  *
                        ;       stacks=0,0 case (,which means do not install the stack.)      *
                        ;       so,after sysparse is done,we have to check if the entered     *
                        ;       values (stack_count,stack_size) are within the actual range,  *
                        ;       (or if "0,0" pair has been entered.)                          *
                        ; input :                                                              *
                        ;   es:si -> parameters in command line.                              *
                        ; output:                                                             *
                        ;   set the variables stack_count,stack_size.                         *
                        ;                                                                     *
                        ; subroutines to be called:                                           *
                        ;   sysinit_parse                                                     *
                        ; logic:                                                              *
                        ; {                                                                   *
                        ;   set di points to stks_parms;                                      *
                        ;   set dx,cx to 0;                                                   *
                        ;   while (end of command line)                                       *
                        ;   { sysinit_parse;                                                  *
                        ;     if (no error) then                                            *
                        ;        { if (cx == 1) then /* first positional = stack count */     *
                        ;             p_stack_count = result_val._$P_picked_val;           *
                        ;          if (cx == 2) then /* second positional = stack size */     *
                        ;             p_stack_size = result_val._$P_picked_val;            *
                        ;        }                                                            *
                        ;     else /*error exit*/                                           *
                        ;        error exit;                                                  *
                        ;   };                                                                *
                        ;   here check p_stack_count,p_stack_size if it meets the condition;  *
                        ;   if o.k.,then set stack_count,stack_size;                          *
                        ;    else error_exit;                                                 *
                        ; };                                                                  *
                        ;************************************************************************

                        tryk:
                                ;if      stacksw

00002C27 80FC4B                 cmp     ah,CONFIG_STACKS ; 'K'
00002C2A 7402                   je      short do_tryk
                        skip_it4:
00002C2C EB79                   jmp     short trys      ; 15/12/2022
                        do_tryk:

                        ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                        ; (SYSINIT:2B33h)
                        ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                        ;%if 0
                        ;ifdef      MULTI_CONFIG
00002C2E E8EB19                 call query_user             ; query the user if config_cmd
00002C31 72F9                   jc   short skip_it4         ; has the CONFIG_OPTION_QUERY bit set
                        ;endif
                        ;%endif     ; 30/10/2022

                                ; 14/12/2022
                                ; ds = cs

00002C33 BF[EA22]               mov     di,stks_parms
00002C36 31C9                   xor     cx,cx
                                ; 03/01/2023
                                ;mov    dx,cx
                        do79:
00002C38 E82C02                 call    sysinit_parse
00002C3B 730B                   jnc     short if79          ; parse error

00002C3D BA[8B51]               mov     dx,badstack         ; "invalid stack parameter"
00002C40 E8111E                 call    print               ;  and show messages and end the search loop.
00002C43 E87402                 call    error_line
                                ;jmp    sr79
                                ; 11/12/2022
00002C46 EB39                   jmp     short sr79
                        if79:
00002C48 83F8FF                 cmp     ax,_$P_RC_EOL       ; end of line?
00002C4B 7412                   je      short en79          ; then end the $endloop

                                ; 14/12/2022
                                ; ds = cs

                                ;;mov   ax,[cs:rv_dword]
                                ;mov    ax,[cs:result_val+_$P_Result_Blk.Picked_Val]
                                ;mov    ax,[rv_dword]
00002C4D A1[1B22]               mov     ax,[result_val+_$P_Result_Blk.Picked_Val]

00002C50 83F901                 cmp     cx,1
00002C53 7505                   jne     short if83

                                ; 14/12/2022
                                ;mov    [cs:p_stack_count],ax
                                ;jmp    short en83
00002C55 A3[1F23]               mov     [p_stack_count],ax
00002C58 EBDE                   jmp     short do79
                        if83:
                                ; 14/12/2022
                                ;mov    [cs:p_stack_size],ax
00002C5A A3[2123]               mov     [p_stack_size],ax
                        en83:
00002C5D EBD9                   jmp     short do79
                        en79:
                                ; 14/12/2022
                                ; ds = cs
00002C5F A1[1F23]               mov     ax,[p_stack_count]
00002C62 09C0                   or      ax,ax
00002C64 741E                   jz      short if87

                                ; 14/12/2022
                                ;cmp    word [p_stack_count],0
                                ;;cmp   word [cs:p_stack_count],0
                                ;je     short if87

                                ; 14/12/2022
00002C66 83F808                 cmp     ax, mincount ; 8
                                ;cmp    word [cs:p_stack_count],mincount ; 8
                                ; 15/12/2022
00002C69 721F                   jb      short en87
00002C6B 833E[2123]20           cmp     word [p_stack_size],minsize ; 32
                                ;cmp    word [cs:p_stack_size],minsize ; 32
```

```
                                    ; 15/12/2022
00002C70 7218                       jb      short en87
                                if94:
                                    ; 14/12/2022
                                    ; ds = cs
                                    ; ax = [p_stack_count]
                                    ;mov    ax,[p_stack_count]
                                    ;;mov   ax,[cs:p_stack_count]
00002C72 A3[8C02]                   mov     [stack_count],ax
                                    ;mov    [cs:stack_count],ax
                                    ;mov    ax,[cs:p_stack_size]
00002C75 A1[2123]                   mov     ax,[p_stack_size]
                                    ;mov    [cs:stack_size],ax
00002C78 A3[8E02]                   mov     [stack_size],ax
                                    ;mov    word [cs:stack_addr],-1      ; stacks= been accepted.
00002C7B C706[9002]FFFF             mov     word [stack_addr],-1
                                sr79:
00002C81 E902F9                     jmp     coff

                                if87:
                                    ; 14/12/2022
00002C84 3906[2123]                 cmp     [p_stack_size],ax ; 0
00002C88 74E8                       je      short if94 ; ax = [p_stack_count] = 0
                                    ;cmp    word [cs:p_stack_size],0
                                    ;je     short if94
                                en87:
                                    ; 15/12/2022
                                    ; ([p_stack_count] is invalid, use default values)
                                    ; 14/12/2022
                                    ; ds = cs
00002C8A C706[8C02]0900             mov     word [stack_count],defaultcount ; 9
00002C90 C706[8E02]8000             mov     word [stack_size],defaultsize ; 128
00002C96 C706[9002]0000             mov     word [stack_addr],0
                                    ;mov    word [cs:stack_count],defaultcount ; 9
                                    ;                                  ; reset to default value.
                                    ;mov    word [cs:stack_size],defaultsize ; 128
                                    ;mov    word [cs:stack_addr],0
00002C9C BA[8B51]                   mov     dx,badstack
00002C9F E8B21D                     call    print
00002CA2 E81502                     call    error_line
00002CA5 EBDA                       jmp     short sr79

                                ; 15/12/2022
                                %if 0
                                    mov     di,stks_parms
                                    xor     cx,cx
                                    ; 03/01/2023
                                    ;mov    dx,cx
                                do79:
                                    call    sysinit_parse
                                    jnc     short if79              ; parse error

                                    mov     dx,badstack             ; "invalid stack parameter"
                                    call    print                   ;  and show messages and end the search loop.
                                    call    error_line
                                    ;jmp    sr79
                                    ; 11/12/2022
                                    jmp     short sr79
                                if79:
                                    cmp     ax,_$P_RC_EOL           ; end of line?
                                    je      short en79              ; then end the $endloop

                                    ;mov    ax,[cs:rv_dword]
                                    mov     ax,[cs:result_val+_$P_Result_Blk.Picked_Val]
                                    cmp     cx,1
                                    jne     short if83

                                    mov     [cs:p_stack_count],ax
                                    jmp     short en83
                                if83:
                                    mov     [cs:p_stack_size],ax
                                en83:
                                    jmp     short do79
                                en79:
                                    cmp     word [cs:p_stack_count],0
                                    je      short if87

                                    cmp     word [cs:p_stack_count],mincount ; 8
                                    jb      short ll88
                                    cmp     word [cs:p_stack_size],minsize ; 32
                                    jnb     short if88
                                ll88:
                                    mov     word [cs:p_stack_count],-1 ; invalid
                                if88:
                                    jmp     short en87

                                    ; 11/12/2022
                                if94:
                                    mov     ax,[cs:p_stack_count]
                                    mov     [cs:stack_count],ax
                                    mov     ax,[cs:p_stack_size]
                                    mov     [cs:stack_size],ax
                                    mov     word [cs:stack_addr],-1      ; stacks= been accepted.
                                sr79:
                                    jmp     coff

                                if87:
                                    cmp     word [cs:p_stack_size],0
                                    je      short en87
                                    mov     word [cs:p_stack_count],-1 ; invalid
                                en87:
                                    cmp     word [cs:p_stack_count],-1 ; invalid?
                                    jne     short if94

                                    mov     word [cs:stack_count],defaultcount ; 9
                                                                    ; reset to default value.
                                    mov     word [cs:stack_size],defaultsize ; 128
                                    mov     word [cs:stack_addr],0

                                    mov     dx,badstack
                                    call    print
                                    call    error_line
                                    jmp     short sr79

                                %endif

                                ; 11/12/2022
                                %if 0
                                if94:
                                    mov     ax,[cs:p_stack_count]
                                    mov     [cs:stack_count],ax
                                    mov     ax,[cs:p_stack_size]
                                    mov     [cs:stack_size],ax
```

```
                                    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:
                                        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"
```

```
36422                                   ;cmp    word [cs:result_val+_$P_Result_Blk.SYNONYM_Ptr],swit_i
36423 00002D7F 813E[1922][9223]         cmp     word [result_val+_$P_Result_Blk.SYNONYM_Ptr],swit_i
36424 00002D85 7507                     jne     short if116
36425                                   ;mov    byte [cs:p_swit_i],1  ; set the flag
36426 00002D87 C606[9823]01             mov     byte [p_swit_i],1
36427 00002D8C EBC3                     jmp     short do110
36428                                   ;;;
36429                                 if116:
36430                                   ; 15/12/2022
36431                                   ;;cmp   word [cs:result_val_swoff],swit_w
36432                                   ;cmp    word [cs:result_val+_$P_Result_Blk.SYNONYM_Ptr],swit_w     ;M063
36433 00002D8E 813E[1922][8623]         cmp     word [result_val+_$P_Result_Blk.SYNONYM_Ptr],swit_w
36434 00002D94 75BB                     jne     short do110                                     ;M063
36435                                   ; 15/12/2022
36436 00002D96 C606[9723]01             mov     byte [p_swit_w],1
36437                                   ;mov    byte [cs:p_swit_w],1                            ;M063
36438 00002D9B EBB4                     jmp     short do110                                     ;M063
36439                                 en110:
36440                                   ; 15/12/2022
36441                                   ; ds = cs
36442 00002D9D 803E[9523]01             cmp     byte [p_swit_k],1
36443                                   ;cmp    byte [cs:p_swit_k],1  ; if /k entered,
36444 00002DA2 1E                       push    ds
36445                                   ;;mov   ax,Bios_Data
36446                                   ;mov    ax,KERNEL_SEGMENT ; 0070h
36447                                   ; 21/10/2022
36448 00002DA3 B87000                   mov     ax,DOSBIODATASEG ; 0070h
36449 00002DA6 8ED8                     mov     ds,ax
36450 00002DA8 750A                     jne     short if117
36451                                   ; 14/04/2024
36452 00002DAA C606[7E04]00             mov     byte [keyrd_func],0 ; 4E5h ; use the conventional keyboard functions
36453                                                               ; BIOSDATA:047Eh for PCDOS 7.1 IBMBIO.COM
36454 00002DAF C606[7F04]01             mov     byte [keysts_func],1 ; 4E6h (for MSDOS 6.21 IO.SYS)
36455                                                                ; BIOSDATA:047Fh for PCDOS 7.1 IBMBIO.COM
36456                                 if117:
36457                                   ; 15/12/2022
36458                                   ; ds <> cs
36459 00002DB4 2EA0[9623]               mov     al,[cs:p_swit_t]                                ;M059
36460 00002DB8 A2[8B04]                 mov     [t_switch],al  ; 4F2h (for MSDOS 6.21 IO.SYS)       ;M059
36461                                   ; 14/04/2024   ; BIOSDATA:048Bh for PCDOS 7.1 IBMBIO.COM
36462 00002DBB 2E803E[9723]00           cmp     byte [cs:p_swit_w],0                            ;M063
36463 00002DC1 740E                     je      short skip_dos_flag                             ;M063
36464 00002DC3 06                       push    es
36465 00002DC4 53                       push    bx
36466 00002DC5 B452                     mov     ah,GET_IN_VARS ; 52h                            ;M063
36467 00002DC7 CD21                     int     21h                                             ;M063
36468                                                   ; DOS - 2+ internal - GET LIST OF LISTS
36469                                                   ; Return: ES:BX -> DOS list of lists
36470                                   ;or     bytes [es:86h],2
36471 00002DC9 26800E860002             or      byte [es:DOS_FLAG_OFFSET],SUPPRESS_WINA20 ; 2       ;M063
36472 00002DCF 5B                       pop     bx
36473 00002DD0 07                       pop     es
36474                                 skip_dos_flag:                                            ;M063
36475 00002DD1 1F                       pop     ds
36476                                 sr110:
36477 00002DD2 E9B1F7                   jmp     coff
36478
36479                                 ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
36480                                 ; (SYSINIT:2D14h)
36481                                 ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
36482                                 ;%if 0
36483
36484                                 tryv:
36485
36486                                 ;ifdef  MULTI_CONFIG
36487                                 ;-----------------------------------------------------------------------
36488                                 ; set command (as in "set var=value<cr/lf>")
36489                                 ;-----------------------------------------------------------------------
36490
36491 00002DD5 80FC56                     cmp     ah,CONFIG_SET  ; 'V'
36492 00002DD8 750F                     jne     short tryn
36493 00002DDA E83F18                   call    query_user      ; query the user if config_cmd
36494 00002DDD 720A                     jc      short tryn     ; has the CONFIG_OPTION_QUERY bit set
36495 00002DDF E83614                   call    copy_envvar     ; copy var at ES:SI to "config_wrkseg"
36496 00002DE2 73EE                     jnc     short sr110    ; no error
36497                                 err:
36498 00002DE4 E8D300                   call    error_line      ; whoops, display error in line XXX
36499 00002DE7 EBE9                     jmp     short sr110     ; jump to coff (to skip to next line)
36500
36501                                 ;-----------------------------------------------------------------------
36502                                 ; numlock command (as in "numlock=on|off")
36503                                 ;-----------------------------------------------------------------------
36504                                 tryn:
36505 00002DE9 80FC4E                   cmp     ah,CONFIG_NUMLOCK  ;'N'
36506                                   ;jne    short tryy
36507                                   ; 14/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
36508 00002DEC 750C                     jne     short tryt
36509
36510 00002DEE E82B18                   call    query_user      ; query the user if config_cmd
36511 00002DF1 7238                     jc      short tryy     ; has the CONFIG_OPTION_QUERY bit set
36512 00002DF3 E8B710                   call    set_numlock
36513 00002DF6 72EC                     jc      short err
36514 00002DF8 EBD8                     jmp     short sr110    ; all done
36515
36516                                 ;endif  ;MULTI_CONFIG
36517
36518                                 ; 14/04/2024 - Retro DOS v5.0 (PCDOS 7.1 IBMBIO.COM)
36519                                 ;-----------------------------------------------------------------------
36520                                 ; dosdata command
36521                                 ;-----------------------------------------------------------------------
36522                                 tryt:
36523                                   ;cmp    ah,54h ; 'T'
36524 00002DFA 80FC54                   cmp     ah,CONFIG_DOSDATA ; 'T' ; PCDOS 7 new config cmd
36525 00002DFD 752C                     jne     short tryy
36526
36527 00002DFF E81A18                   call    query_user
36528 00002E02 7227                     jc      short tryy
36529
36530 00002E04 BF[C823]                 mov     di,dosdata_parms
36531 00002E07 31C9                     xor     cx,cx
36532                                   ; 14/04/2024 - Retro DOS v5.0
36533                                   ;mov    dx,cx ; 0
36534                                 do120:
36535 00002E09 E85B00                   call    sysinit_parse
36536 00002E0C 7303                     jnc     short if120
36537
36538                                   ;call   badparm_p
36539                                   ;jmp    short en120
36540                                   ; 14/04/2024 - Retro DOS v5.0
36541 00002E0E E945FF                   jmp     badparm_p_coff
36542                                 if120:
36543                                   ;cmp    ax,0FFFFh
36544 00002E11 83F8FF                   cmp     ax,_$P_RC_EOL ; -1 ; end of line?
36545 00002E14 7422                     jz      short en120
```

```
36546 00002E16 803E[1822]01              cmp     byte [result_val_itag],1 ; tag 1 (UMB)
36547                                                    ; [result_val+_$P_Result_Blk.Item_Tag]
36548 00002E1B 7507                      jnz     short if121
36549 00002E1D C606[6E03]01              mov     byte [dosdata_umb],1 ; DOSDATA=UMB (1) NOUMB (0)
36550                                    ;jmp    short sr120
36551                                    ; 14/04/2024
36552 00002E22 EBE5                      jmp     short do120
36553                                  if121:
36554 00002E24 C606[6E03]00              mov     byte [dosdata_umb],0 ; DOSDATA=UMB (1) NOUMB (0)
36555                                  sr120:
36556 00002E29 EBDE                      jmp     short do120
36557                                    ; 14/04/2024
36558                                  ;en120:
36559                                    ;jmp    coff
36560
36561                                  ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
36562                                  ;--------------------------------------------------------------------------
36563                                  ; comment= do nothing. just decrease chrptr,and increase count for correct
36564                                  ;          line number
36565                                  ;--------------------------------------------------------------------------
36566
36567                                    ; 31/12/2022
36568                                  tryy:
36569 00002E2B 80FC59                    cmp     ah,CONFIG_COMMENT ; 'Y'
36570 00002E2E 750B                      jne     short try0
36571
36572                                  donothing:
36573                                    ; 15/12/2022
36574                                    ; ds = cs
36575 00002E30 FF0E[5A03]                dec     word [chrptr]
36576 00002E34 FF06[5603]                inc     word [count]
36577                                    ; 02/11/2022
36578                                    ;dec    word [cs:chrptr]
36579                                    ;inc    word [cs:count]
36580                                  en120:              ; 14/04/2024
36581 00002E38 E94BF7                    jmp     coff
36582
36583                                  ;--------------------------------------------------------------------------
36584                                  ; rem command
36585                                  ;--------------------------------------------------------------------------
36586
36587                                  try0:                           ; do nothing with this line.
36588 00002E3B 80FC30                      cmp     ah,CONFIG_REM ; '0'
36589 00002E3E 74F0                        je      short donothing
36590
36591                                  ;%endif    ; 30/10/2022
36592
36593                                  ; 30/10/2022
36594                                  ; (MSSOS 5.0 IO.SYS - SYSINIT:29D7h)
36595
36596                                  ;--------------------------------------------------------------------------
36597                                  ; bogus command
36598                                  ;--------------------------------------------------------------------------
36599
36600                                  tryz:
36601 00002E40 80FCFF                          cmp     ah,0FFh           ;null command? (BUGBUG - who sets FFh anyway?)
36602                                    ; 31/12/2022
36603 00002E43 74EB                      je      short donothing
36604                                    ; 02/11/2022
36605                                    ;je     short tryz_donothing
36606
36607 00002E45 FF0E[5A03]                dec     word [chrptr]
36608 00002E49 FF06[5603]                inc     word [count]
36609 00002E4D EB37                      jmp     short badop
36610
36611                                  ; 31/12/2022
36612                                  ;   ; 02/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
36613                                  ;tryz_donothing:
36614                                  ;   jmp     donothing
36615
36616                                  ; 07/04/2019 - Retro DOS v4.0
36617
36618                                  ;--------------------------------------------------------------------------------
36619
36620                                  ; 31/12/2022 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
36621                                  ; (SYSINIT:2D5Dh)
36622
36623                                  ; 11/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
36624
36625                                  ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
36626                                  ;
36627                                  ;***CheckProtmanArena -- special hack for adjusting alloclim with Protman$
36628                                  ;
36629                                  ;   adjusts alloclim if Protman$ reduced our arena through a manual hack.
36630                                  ;
36631                                  CheckProtmanArena:
36632                                      ; 08/09/2023
36633                                      ; ds = cs
36634 00002E4F 06                        push    es
36635                                    ;mov    ax,[cs:area]   ; get our arena header
36636 00002E50 A1[6803]                  mov     ax,[area] ; 08/09/2023
36637 00002E53 48                        dec     ax
36638 00002E54 8EC0                      mov     es,ax
36639                                    ;add    ax,[es:ARENA.SIZE]
36640 00002E56 2603060300                add     ax,[es:3]      ; find end of arena
36641 00002E5B 40                        inc     ax
36642                                    ; 08/09/2023
36643 00002E5C 3B06[A502]                cmp     ax,[ALLOCLIM]
36644                                    ;cmp    ax,[cs:ALLOCLIM] ; is it less than alloclim?
36645 00002E60 7703                      ja      short CheckProtmanDone
36646
36647                                    ;mov    [cs:ALLOCLIM],ax ; reduce alloclim then
36648                                    ; 08/09/2023
36649 00002E62 A3[A502]                  mov     [ALLOCLIM],ax
36650                                  CheckProtmanDone:
36651 00002E65 07                        pop     es
36652 00002E66 C3                        retn
36653
36654                                  ;--------------------------------------------------------------------------------
36655
36656                                  sysinit_parse:
36657
36658                                  ;--------------------------------------------------------------------------------
36659                                  ;set up registers for sysparse
36660                                  ;in)es:si -> command line in confbot
36661                                  ;   di -> offset of the parse control definition.
36662                                  ;
36663                                  ;out)      calls sysparse.
36664                                  ;   carry will set if parse error.
36665                                  ;   *** the caller should check the eol condition by looking at ax
36666                                  ;   *** after each call.
36667                                  ;   *** if no parameters are found,then ax will contain a error code.
36668                                  ;   *** if the caller needs to look at the synomym@ of the result,
36669                                  ;   ***  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
00002F9D B110                       ;mov    cx,MAXUMB ; 16              ;Okay... for each entry
                                    ; 01/01/2033
00002F9F BF[1024]                   mov     cl,MAXUMB ; 16
00002FA2 F3AB                       mov     di,UmbSize                  ;on the UmbSize array,
                                    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.
```

```
37538
37539 00003023 AC                      pv20:       lodsb                   ; Just read 'S' or 'L', hopefully
37540                                      ;toUpper al             ; So we make it upper-case, and...
37541 00003024 24DF                        and     al,0DFh
37542                                      ;cmp    al,'S'
37543 00003026 3C53                        cmp     al,optS         ; just read 'S'?
37544 00003028 750D                        jne     short pv30
37545
37546                                      ;call   incArgc         ; If it's /S, it's another arg for LH to skip.
37547 0000302A 2EFE06[3224]                inc     byte [cs:fm_argc] ; 19/04/2019
37548
37549                                      ;putdata fUmbTiny,1     ; /S, so ES:SI=="  /L..." or " module opts", or
37550                                      ;
37551                                      ;push   es
37552                                      ;push   cs
37553                                      ;pop    es
37554                                      ;mov    [es:fUmbTiny],1
37555                                      ;pop    es
37556
37557 0000302F 2EC606[FC23]01               mov     byte [cs:fUmbTiny],1
37558
37559 00003035 EBDE                         jmp     short pv10      ; possibly even "/L...".
37560
37561                                  pv30:       ;cmp    al,'L'
37562 00003037 3C4C                        cmp     al,optL         ; If it's not 'L' either, then 'tis a bad
37563 00003039 750D                        jne     short pvE1              ; switch!
37564
37565                                      ;call   incArgc         ; If it's /L, it's another arg for LH to skip.
37566 0000303B 2EFE06[3224]                inc     byte [cs:fm_argc] ; 19/04/2019
37567
37568 00003040 E80E00                       call    parseL
37569 00003043 73D0                        jnc     short pv10      ; If no carry, go back and look for more
37570
37571 00003045 4E                          dec     si              ; Else, back up and exit.
37572 00003046 EB03                        jmp     short pvErr     ; AX has already been set by parseL
37573
37574                                  pvE1:       ;mov    ax,3
37575 00003048 B80300                      mov     ax,PV_InvSwt    ; Unrecognized switch passed
37576 0000304B 4E                      pvErr:      dec     si
37577 0000304C 4E                          dec     si
37578 0000304D F9                          stc
37579                                  pvX:;popreg <es, ds, di>
37580 0000304E 07                          pop     es
37581 0000304F 1F                          pop     ds
37582                                      ; 01/01/2023
37583                                      ;pop    di ; * ; (not required) ; 01/01/2023
37584 00003050 C3                          retn
37585
37586                                  ; ---------------------------------------------------------------------------
37587                                  ;*** parseL - parses ":nnnn[,nnnn][;nnnn[,nnnn]]*" for ParseVar
37588                                  ; ---------------------------------------------------------------------------
37589                                  ; ENTRY:    ES:SI points to colon
37590                                  ; EXIT:     ES:SI points to first character not parsed
37591                                  ; ERROR:    Carry set; rewind three characters and return (see ParseVar)
37592                                  ; USES:     ES:SI, flags, AX, CX, DX, variables in highvar.inc
37593                                  ; ---------------------------------------------------------------------------
37594                                  ; If the string here is terminated with anything other than whitespace or a
37595                                  ; switchchar (perhaps it's /S or another /L:... ), then we return with carry
37596                                  ; set, indicating that they've screwed up the syntax.  The 3-character rewind
37597                                  ; makes sure the app /L: is reported as being the culprit.
37598                                  ; ---------------------------------------------------------------------------
37599
37600                                  parseL:
37601 00003051 AC                          lodsb
37602 00003052 3C3A                        cmp     al,':'          ; Make sure they did /L:
37603 00003054 754E                        jne     short plE1      ; If they didn't, return with carry set.
37604
37605                                  ; -----------------------------------------
37606                                  ; PL10--ES:SI = a UMB number, after /L: or ;
37607                                  ; -----------------------------------------
37608
37609 00003056 E8DB00                  pl10:       call    GetXNum         ; After this, 'tis ",size" or ";umb" or " mod"
37610 00003059 724F                        jc      short plE2      ; And error if it's a bad number.
37611 0000305B E89D01                      call    convUMB         ; Convert any address to a UMB number
37612
37613 0000305E 88C1                        mov     cl,al           ; Remember the UMB number
37614 00003060 E87600                      call    stowUMB         ; Mark this UMB # as used;
37615 00003063 7245                        jc      short plE2      ; If it was already marked, it'll error
37616
37617                                      ;call   incArgc         ; Each UMB number is another arg for LH to skip
37618 00003065 2EFE06[3224]                inc     byte [cs:fm_argc] ; 08/04/2019 - Retro DOS v4.0
37619
37620 0000306A AC                          lodsb
37621 0000306B 3C3B                        cmp     al,';'          ; Did "umb;" ?
37622 0000306D 74E7                        je      short pl10      ; Yep: go back and get another UMB.
37623
37624 0000306F E84900                      call    isWhite         ; Did "umb " ?
37625 00003072 743B                        jz      short plX       ; Yep: return (it'll go back to whitespace)
37626
37627 00003074 E83900                      call    isEOL           ; Did "umb" ?
37628 00003077 7435                        jz      short plSwX     ; If so, backup and exit like everything's ok
37629
37630                                      ;cmp    al,'/'
37631 00003079 3C2F                        cmp     al,SWTCH        ; Did "umb/" ? (as in, "/L:1,100;2/S")
37632 0000307B 7431                        je      short plSwX     ; If so, back up ES:SI one character and return
37633
37634 0000307D 3C2C                        cmp     al,','          ; Did "umb," ?
37635 0000307F 7523                        jne     short plE1      ; Just what the heck DID they do? Return error.
37636
37637                                  ; --- Read a size -----------------------------------------------------------
37638
37639 00003081 E8B000                      call    GetXNum         ; Stop on "size;" or "size " or anything else
37640 00003084 721E                        jc      short plE1      ; And error if it's a bad size.
37641
37642 00003086 E81601                      call    toPara          ; Convert from bytes to paragraphs
37643
37644 00003089 E8CE01                      call    stowSiz         ; CL still has the UMB number for this routine
37645
37646                                      ;call   incArgc         ; Each UMB size is another arg for LH to skip
37647 0000308C 2EFE06[3224]                inc     byte [cs:fm_argc] ; 08/04/2019 - Retro DOS v4.0
37648
37649 00003091 AC                          lodsb
37650 00003092 3C3B                        cmp     al,';'          ; They did "umb,size;", so get another UMB.
37651 00003094 74C0                        je      short pl10      ;
37652
37653 00003096 E82200                      call    isWhite         ; Did it end with whitespace?
37654 00003099 7414                        jz      short plX       ; If so, we're done here--go back.
37655
37656 0000309B E81200                      call    isEOL           ; Did they do "umb,size" and end??? (stupid)
37657 0000309E 740E                        jz      short plSwX     ; If so, backup and exit like everything's ok
37658
37659                                      ;cmp    al,'/'
37660 000030A0 3C2F                        cmp     al,SWTCH        ; Did they do "umb,size/" ?
37661 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
000030D4 8826[FF23]                 mov     [UmbLoad],ah ; 0
                                umu10:
                                    ;pop    es
                                    ;pop    di
                                    ;pop    bx
                                    ;pop    ax
000030D8 C3                         retn

                                ; ---------------------------------------------------------------------------
                                ;*** stowUMB - marks a given UMB as used, if it hasn't been so marked before
                                ;           -- accepts a UMB # in AL, and makes sure it hasn't yet been
                                ; listed in the /L:... chain. If it's the first one specified, it sets UmbLoad
                                ; to that UMB #... and in any case, it marks the UMB as specified.
                                ; ---------------------------------------------------------------------------
                                ; ENTRY:    AL contains UMB number, as specified by the user
                                ; EXIT:     None
                                ; ERROR:    Carry set if UMB # is less than 0 or >= MAXUMB (see highvar.inc)
                                ; USES:     AX, Flags, variables in highvar.inc
                                ; ---------------------------------------------------------------------------

                                    ; 01/01/2023 - Retro DOS v4.2
                                stowUMB:
000030D9 3C10                       cmp     al,MAXUMB ; 16
000030DB 7202                       jb      short su10
000030DD F9                         stc
000030DE C3                         retn                    ; Ooops-- UMB>=MAXUMB
                                su10:
                                    ; 01/01/2023
                                    ;push   bx
                                    ;push   di
                                    ;push   si
                                    ;push   ds
                                    ;push   es
                                    ;push   cs
                                    ;pop    es
                                    ;push   cs
                                    ;pop    ds

                                    ; 01/01/2023
                                    ; ds <> cs
                                    ;cmp    byte [cs:UmbLoad],0FFh
000030DF 2E803E[FF23]FF             cmp     byte [cs:UmbLoad],UNSPECIFIED
                                                            ; If this, we haven't been here before
000030E5 7504                       jne     short su20
000030E7 2EA2[FF23]                 mov     [cs:UmbLoad],al       ; So remember this UMB as the load UMB slot.

                                    ;;cmp   byte [UmbLoad],0FFh
                                    ;cmp    byte [UmbLoad],UNSPECIFIED ; If this, we haven't been here before
                                    ;jne    short su20
                                    ;mov    [UmbLoad],al   ; So remember this UMB as the load UMB slot.
                                su20:
000030EB 08C0                       or      al,al           ; If they gave UMB 0, there's really nothing
000030ED 740E                       jz      short su30      ; that we should do here.

                                    ;mov    bl,al
                                    ;xor    bh,bh
                                    ;mov    ax,1            ; Now, AX = 1, and BX = UMB Number
                                    ; 01/01/2023
000030EF 30E4                       xor     ah,ah
000030F1 89C3                       mov     bx,ax
000030F3 B001                       mov     al,1

                                    ;xchg   [es:bx+UmbUsed],al
                                    ; 01/01/2023
000030F5 2E8687[0024]               xchg    [cs:bx+UmbUsed],al

                                    ;or     al,al           ; If it was already 1, then al==1... and that
                                    ;jz     short su30      ; means an error.
                                    ;
                                    ;stc                    ; OOOPS!  This one's been used before.  :(

                                    ; 01/01/2023
000030FA 3C01                       cmp     al,1
000030FC F5                         cmc     ; if al > 0 -> cf = 1
                                su30:
                                    ; 01/01/2023
                                    ;pop    es
                                    ;pop    ds
                                    ;pop    si
                                    ;pop    di
                                    ;pop    bx
000030FD C3                         retn

                                ; 08/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
                                %if 0
                                ; ---------------------------------------------------------------------------
                                ;*** 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

                                    mov     bl,cl                   ; Now bl==UMB number, AX==size
                                    mov     bh,0                    ;     bx==UMB number, AX==size
                                    shl     bl,1                    ;     bx==offset into array, AX=size
                                    ;mov    [es:bx+UmbSize],ax      ; Store the size
                                    ; 01/01/2023
                                    mov     [cs:bx+UmbSize],ax      ; Store the size

                                    ; 01/01/2023
                                    ;pop    es
                                    ;;pop   di ; ?
                                    ;pop    bx

                                    retn
                                %endif

                                ; ---------------------------------------------------------------------------
                                ;*** toDigit - converts a character-digit to its binary counterpart
```

```
                                ;           -- 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.
                                    ;jmp    short gxn20
0000317A 73EC                       jnc     short gxn20
                                gxnE:
                                    ;stc                    ; In this case, we need to set the carry
0000317C EB02                       jmp     short gxnX      ; and leave--there were no digits given.
                                gxnQ:
0000317E 4E                         dec     si              ; Don't read in the offensive character.
0000317F F8                         clc                     ; And clear carry, so they know it's okay.
                                gxnX:
                                    ; 01/01/2023
                                    ;pop    ds
00003180 59                         pop     cx ; *
                                    ;pop    bx
00003181 C3                         retn

                                ; ------------------------------------------------------------------------------
                                ;*** mul32 - multiplies the number in DX:AX by gnradix
                                ; ------------------------------------------------------------------------------
                                ; ENTRY:   DX:AX = the number to be multiplied, BX = 0, gnradix = multiplier
                                ; EXIT:    DX:AX has been multiplied by gnradix if carry clear; BX still 0
                                ; ERROR:   Carry set if number was too large
                                ; USES:    Flags, AX, DX
                                ; ------------------------------------------------------------------------------

                                mul32:
00003182 50                         push    ax              ; DX=old:hi, AX=old:lo, TOS=old:lo, BX=0
00003183 89D0                       mov     ax,dx           ; DX=old:hi, AX=old:hi, TOS=old:lo, BX=0
00003185 2EF726[FE30]               mul     word [cs:gnradix] ; DX=?, AX=new:hi, TOS=old:lo, BX=0
0000318A 7211                       jc      short m32E      ; Too big?

0000318C 89C2                       mov     dx,ax           ; DX=new:hi, AX=new:hi, TOS=old:lo, BX=0
0000318E 58                         pop     ax              ; DX=new:hi, AX=old:lo, TOS=orig, BX=0

0000318F 87DA                       xchg    dx,bx           ; DX=0, AX=old:lo, TOS=orig, BX=new:hi
00003191 2EF726[FE30]               mul     word [cs:gnradix] ; DX=carry,  AX=new:lo, TOS=orig, BX=new:hi
00003196 87DA                       xchg    dx,bx           ; DX=new:hi, AX=new:lo, TOS=orig, BX=carry
00003198 01DA                       add     dx,bx           ; DX=new:hi, AX=new:lo, TOS=orig, BX=carry
0000319A 31DB                       xor     bx,bx           ; DX=new:hi, AX=new:lo, TOS=orig, BX=0
0000319C C3                         retn
                                m32E:
0000319D 58                         pop     ax
0000319E C3                         retn

                                ; ------------------------------------------------------------------------------
                                ;*** toPara - divides DX:AX by 16; result in AX only (discards extra DX data)
                                ; ------------------------------------------------------------------------------
                                ; ENTRY:   DX:AX = the number to be divided
                                ; EXIT:    Interpereting DX:AX as bytes, AX=paragraph equivalent, 0xFFFF max
                                ; ERROR:   None
                                ; USES:    Flags, AX, DX
                                ; ------------------------------------------------------------------------------
                                ; Note: The 386 has a 32-bit SHR, which would work perfectly for this... but we
                                ;       can't ensure a 386 host machine. Sorry.
                                ; ------------------------------------------------------------------------------

                                    ; 01/01/2023 - Retro DOS v4.2
                                toPara:
0000319F 51                         push    cx              ; DX:AX=HHHH hhhh hhhh hhhh:LLLL llll llll llll

000031A0 B104                       mov     cl,4            ;
000031A2 D3E8                       shr     ax,cl           ; DX:AX=HHHH hhhh hhhh hhhh:0000 LLLL llll llll
000031A4 92                         xchg    ax,dx           ; DX:AX=0000 LLLL llll llll:HHHH hhhh hhhh hhhh
000031A5 B10C                       mov     cl,12
000031A7 D3E0                       shl     ax,cl           ; DX:AX=0000 LLLL llll llll:hhhh 0000 0000 0000
000031A9 09D0                       or      ax,dx           ;    AX=hhhh LLLL llll llll

000031AB 59                         pop     cx
000031AC C3                         retn

                                ; ------------------------------------------------------------------------------
                                ;*** UmbHead - 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
                                ; ------------------------------------------------------------------------------
                                ; Early in the boot-cycle, the pointer used to obtain this value isn't set up;
                                ; to be precise, before a UMB provider is around. In this event, the pointer
                                ; is always set to 0xFFFF; it changes once a provider is around. On most
                                ; machines (all of 'em I've seen), it changes to 0x9FFF at that point.
                                ; ------------------------------------------------------------------------------

                                    ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                UmbHead:
                                    ; 13/05/2019 (because of callers, pushs & pops are not needed here)

                                    ;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)

000031AD B452                       mov     ah,GET_IN_VARS      ; Call int 21h, function 52h...
000031AF CD21                       int     21h

000031B1 26A18C00                   mov     ax,[es:DOS_UMB_HEAD]  ; And read what's in ES:[008C]

                                    ; 01/01/2023
000031B5 83F8FF                     cmp     ax,0FFFFh
000031B8 F5                         cmc
                                    ; if AX=0FFFFh -> CF=1
000031B9 C3                         retn

                                ; 01/01/2023
                                ;%if 0
                                ;   cmp     ax,0FFFFh
                                ;   je      short uhE           ; If it's 0xFFFF, it's an error...
                                ;
                                ;   clc                         ; Else, it isn't (CLC done by prev cmp)
                                ;   ;jmp    short uhX
                                ;   ; 12/12/2022
                                ;   retn
                                ;uhE:
                                ;   stc
                                ;uhX:
                                ;   ;pop    bx ; *
                                ;   ;pop    es
                                ;   ;pop    ds ; ?
                                ;   ;pop    si ; ?
                                ;   retn
```

```
                                ;%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
```

```
38282
38283                                   ; ----------------------------------------------------------------------------
38284                                   ;*** convUMB - checks after GetXNum to convert an address to a UMB number
38285                                   ;           -- if GetXNum read a hex number, we interperete that as a segment
38286                                   ; address rather than a UMB number... and use that address to look up a UMB.
38287                                   ; This routine checks for that condition and calls AddrToUmb if necessary.
38288                                   ; ----------------------------------------------------------------------------
38289                                   ; ENTRY:  AX contains a UMB number or segment, gnradix has been set by GetXNum
38290                                   ; EXIT:   AX will contain a UMB number
38291                                   ; ERROR:  None
38292                                   ; USES:   Flags, AX
38293                                   ; ----------------------------------------------------------------------------
38294
38295                                       ; 01/01/2023 - Retro DOS v4.2
38296                                   convUMB:
38297 000031FB 2E833E[FE30]10             cmp     word [cs:gnradix],16
38298 00003201 7507                       jne     short cu10      ; If it didn't read in hex, it's not an address
38299 00003203 E8C4FF                     call    AddrToUmb       ; Else, convert the address to a UMB number
38300                                     ;cmp    ax,0FFFFh
38301                                     ;jne    short cu10
38302                                     ;inc    ax              ; If too high, ignore it (make it conventional)
38303                                     ; 01/01/2023
38304 00003206 40                         inc     ax
38305 00003207 7401                       jz      short cu10      ; If too high, ignore it (make it conventional)
38306 00003209 48                         dec     ax
38307                                   cu10:
38308 0000320A C3                         retn
38309
38310                                   ; 01/01/2023 - Retro DOS v4.2
38311                                   ;%if 0
38312                                   ;
38313                                   ;; ----------------------------------------------------------------------------
38314                                   ;;*** setUMBs - links umbs and sets allocation strategy for a load
38315                                   ;;           -- if LoadHigh, the allocation strategy MAY be LOW_FIRST instead
38316                                   ;; of the usual HIGH_FIRST. See the code.
38317                                   ;; ----------------------------------------------------------------------------
38318                                   ;; ENTRY:  None
38319                                   ;; EXIT:   None
38320                                   ;; ERROR:  None
38321                                   ;; USES:   Flags, fm_umb, fm_strat
38322                                   ;; ----------------------------------------------------------------------------
38323                                   ;
38324                                   ;setUMBs:
38325                                   ;   push    ax
38326                                   ;   push    bx
38327                                   ;   call    fm_link
38328                                   ;   pop     bx
38329                                   ;   pop     ax
38330                                   ;   retn
38331                                   ;
38332                                   ;%endif
38333
38334                                   ; 18/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
38335                                   ; loadLow subroutine is not used anywhere of IO.SYS 6.22 (& 5.0)
38336                                   %if 0
38337
38338                                   ; ----------------------------------------------------------------------------
38339                                   ;*** loadLow - returns AL==0 if UMB0 == 0, else AL==1
38340                                   ; ----------------------------------------------------------------------------
38341                                   ; ENTRY:  None
38342                                   ; EXIT:   AL==0 if mem strategy should be set to LOW_FIRST, else AL==1
38343                                   ;         Carry set if UMB0 not specified (_NOT_ an error)
38344                                   ; ERROR:  None
38345                                   ; USES:   Flags, fm_strat, fm_umb
38346                                   ; ----------------------------------------------------------------------------
38347                                   ; We want to set the memory strategy to LOW_FIRST if the user specified a
38348                                   ; load UMB, and it is 0. That 0 can be either from the user having _specified_
38349                                   ; zero (/L:0;...), or from having specified a too-big min size (/L:1,99999999)
38350                                   ; such that the load UMB is too small, and shouldn't be used.
38351                                   ; ----------------------------------------------------------------------------
38352
38353                                   loadLow:
38354                                       ;push   ds
38355                                       ;push   cs              ; Point DS into appropriate data segment
38356                                       ;pop    ds
38357
38358                                       ;mov    al,[UmbLoad]
38359                                       mov     al,[cs:UmbLoad]
38360                                       cmp     al,UNSPECIFIED ; 0FFh, -1
38361                                       jne     short ll10
38362
38363                                       stc
38364                                   ll15:
38365                                       mov     al,1            ; Return with AL==1 && STC if no UMBs specified
38366                                       ;stc
38367                                       ;jmp    short llX
38368                                       retn
38369                                   ll10:
38370                                       or      al,al           ; AL=the load UMB: Is it == 0?
38371                                       ;jz     short llX       ; Yep... CF==0 (from OR) && AL=0, so just exit
38372
38373                                       jnz     short ll15      ; 09/04/2019 - Retro DOS v4.0
38374                                       retn
38375
38376                                       ;mov    al,1
38377                                       ;clc
38378                                   ;llX:
38379                                       ;pop    ds              ; Return DS to where it was
38380                                       ;retn
38381
38382                                   %endif
38383
38384                                   ; ----------------------------------------------------------------------------
38385                                   ;*** HideUMBs - links UMBs and hides upper-memory as appropriate
38386                                   ; ----------------------------------------------------------------------------
38387                                   ; ENTRY:  None
38388                                   ; EXIT:   None
38389                                   ; ERROR:  None
38390                                   ; USES:   Flags, fm_strat, fm_umb
38391                                   ; ----------------------------------------------------------------------------
38392
38393                                       ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
38394                                       ; (SYSINIT:322Fh)
38395                                   HideUMBs:
38396                                       ; 01/01/2023
38397                                       ;push   ax
38398                                       ;push   cx
38399                                       ;push   ds
38400 0000320B 06                         push    es
38401
38402                                       ; 01/01/2023
38403                                       ; ds = cs
38404
38405 0000320C E86E02                     call    UmbTest         ; See if we REALLY linked in anything...
```

```
38406 0000320F 7232                     jc      short husX      ; ...if not, there's nothing for us to do.
38407
38408 00003211 E892FD                   call    FixMem          ; Concatenate adjacent free MCBs in upper mem
38409
38410                                   ;call   setUMBs         ; Link UMBs and set memory-allocation strategy
38411                                   ; 01/01/2023
38412 00003214 E8DCFD                   call    fm_link
38413
38414                                   ;putdata fInHigh,1      ; Remember that we're now running high
38415                                   ;mov    byte [cs:fInHigh],1
38416                                   ; 01/01/2023
38417 00003217 C606[FB23]01             mov     byte [fInHigh],1
38418
38419                                   ;call   GetLoadUMB      ; See if they gave us a list to leave free
38420                                   ;mov    al,[cs:UmbLoad] ; 09/04/2019 - Retro DOS v4.0
38421                                   ; 01/01/2023
38422 0000321C A0[FF23]                 mov     al,[UmbLoad]
38423
38424 0000321F 3CFF                     cmp     al,UNSPECIFIED ; If they didn't,
38425 00003221 7420                     je      short husX      ; then we shouldn't do this loop:
38426
38427 00003223 31C9                     xor     cx,cx
38428
38429                                 ; ------------------------------------------------
38430                                 ; HUS10-CX - UMB number (after inc, 1==first UMB)
38431                                 ; ------------------------------------------------
38432
38433 00003225 41                    hus10:      inc     cx              ; For each UMB:
38434                                    ; 01/01/2023
38435 00003226 80F910                   cmp     cl,MAXUMB
38436                                   ;cmp    cx,MAXUMB ; 16
38437 00003229 730E                     jae     short hus20
38438
38439 0000322B 88C8                     mov     al,cl           ; (stopping as soon as we're outside of the
38440 0000322D 06                       push    es
38441 0000322E E8A200                   call    findUMB         ; valid range of UMBs)
38442 00003231 07                       pop     es              ; push/pop: trash what findumb finds. :-)
38443 00003232 7205                     jc      short hus20
38444
38445                                   ; 02/01/2023
38446                                   ;push   cx ; *
38447 00003234 E84F01                   call    _hideUMB_       ; hide what we need to hide.
38448                                   ;pop    cx ; *
38449
38450 00003237 EBEC                     jmp     short hus10
38451                                hus20:
38452                                   ;call   GetLoadUMB      ; Now check if they offered /L:0
38453                                   ; 01/01/2023
38454                                   ; ds = cs
38455                                   ;mov    al,[UmbLoad]
38456                                   ;;mov   al,[cs:UmbLoad] ; 09/04/2019 - Retro DOS v4.0
38457 00003239 800E[FF23]00             or      byte [UmbLoad],0
38458                                   ;or     al,al           ; --Is the load UMB 0? (-1==unspecified)
38459 0000323E 7503                     jnz     short husX      ; If not, we're done.
38460
38461 00003240 E86802                   call    hl_unlink       ; If so, however, fix UMBs and strategy.
38462                                husX:
38463 00003243 07                       pop     es
38464                                   ; 01/01/2023
38465                                   ;pop    ds
38466                                   ;pop    cx
38467                                   ;pop    ax
38468 00003244 C3                       retn
38469
38470                                ; ---------------------------------------------------------------------------
38471                                ;*** GetLoadUMB - Returns the load UMB number in AL (-1 if not specified)
38472                                ; ---------------------------------------------------------------------------
38473                                ; ENTRY:  None
38474                                ; EXIT:   AL == load UMB
38475                                ; ERROR:  None
38476                                ; USES:   Flags, AX
38477                                ; ---------------------------------------------------------------------------
38478
38479                                ;GetLoadUMB:
38480                                ;   ;getdata al, UmbLoad
38481                                ;   push    ds
38482                                ;   push    cs
38483                                ;   pop     ds
38484                                ;   mov     al,[UmLoad]
38485                                ;   pop     ds
38486                                ;   retn
38487
38488                                ; ---------------------------------------------------------------------------
38489                                ;*** GetLoadSize - Returns the load UMB minimum size (0 if not specified)
38490                                ; ---------------------------------------------------------------------------
38491                                ; ENTRY:  None
38492                                ; EXIT:   AX == load UMB minimum size
38493                                ; ERROR:  None
38494                                ; USES:   Flags, AX
38495                                ; ---------------------------------------------------------------------------
38496
38497                                ; 08/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
38498                                %if 0
38499                                    ; 01/01/2023 - Retro DOS v4.2
38500                                GetLoadSize:
38501                                    ; 09/04/2019 - Retro DOS v4.0
38502                                    ;mov    al,[cs:UmbLoad]
38503                                    ; 01/01/2023
38504                                    ; ds = cs
38505                                    mov     al,[UmbLoad]
38506                                    ;jmp    short GetSize
38507
38508                                    ;push   bx
38509                                    ;;push  si
38510                                    ;push   ds
38511                                    ;push   cs
38512                                    ;pop    ds
38513
38514                                    ;mov    al,[UmbLoad]
38515
38516                                    ;xor    ah,ah                   ;    ax==UMB
38517                                    ;mov    bx,UmbSize              ;    bx==array
38518                                    ;shl    al,1                     ;    ax==offset
38519                                    ;;add   ax,bx                   ;    ax==element index
38520                                    ;;mov   si,ax                   ; ds:si==element index
38521
38522                                    ;;lodsw                         ;    hh
38523
38524                                    ;add    bx,ax
38525                                    ;mov    ax,[bx]
38526
38527                                    ;pop    ds
38528                                    ;;pop   si
38529                                    ;pop    bx
```

```
                                    ;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
```

```
38654                                   ;   retn
38655
38656                                       ; 08/09/2023
38657                                       ;;jmp   stowSiz
38658                                       ;jmp    short stowSiz
38659
38660                                   ; 08/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
38661                                   %if 1
38662                                   ; ---------------------------------------------------------------------------
38663                                   ;*** stowSiz - marks a given UMB as having a given minimum size
38664                                   ; ---------------------------------------------------------------------------
38665                                   ; ENTRY:    CL contains UMB number, AX contains size
38666                                   ; EXIT:     None
38667                                   ; ERROR:    None
38668                                   ; USES:     AX, DX, Flags, variables in highvar.inc
38669                                   ; ---------------------------------------------------------------------------
38670
38671                                   ; 13/05/2019
38672
38673                                       ; 01/01/2023 - Retro DOS v4.2
38674                                   stowSiz:
38675                                       ; 01/01/2023
38676                                       ;push   bx
38677                                       ;;push  di ; ?
38678                                       ;push   es
38679
38680                                       ;push   cs
38681                                       ;pop    es
38682
38683 0000325A 88CB                        mov     bl,cl                   ; Now bl==UMB number, AX==size
38684 0000325C B700                        mov     bh,0                    ;     bx==UMB number, AX==size
38685 0000325E D0E3                        shl     bl,1                    ;     bx==offset into array, AX=size
38686                                      ;mov    [es:bx+UmbSize],ax      ; Store the size
38687                                      ; 01/01/2023
38688 00003260 2E8987[1024]                mov     [cs:bx+UmbSize],ax      ; Store the size
38689
38690                                      ; 01/01/2023
38691                                      ;pop    es
38692                                      ;;pop   di ; ?
38693                                      ;pop    bx
38694
38695 00003265 C3                          retn
38696                                   %endif
38697
38698                                   ; ---------------------------------------------------------------------------
38699                                   ;*** hideUMB - marks as HIDDEN all FREE elements in UMB passed as AL
38700                                   ; ---------------------------------------------------------------------------
38701                                   ; ENTRY:    AL must indicate a valid UMB; 0==conv && is invalid.
38702                                   ; EXIT:     None; free elements in UMB marked as hidden
38703                                   ; ERROR:    None
38704                                   ; USES:     Flags
38705                                   ; ---------------------------------------------------------------------------
38706
38707                                       ; 01/01/2023 - Retro DOS v4.2
38708                                   hideUMB:
38709                                       ; 02/01/2023
38710 00003266 52                          push    dx ; (*)
38711                                      ; 01/01/2023
38712                                      ;push   ax
38713 00003267 06                          push    es
38714
38715 00003268 E86800                      call    findUMB ; (*)   ; Returns with carry if err, else ES == MCB
38716 0000326B 7224                        jc      short huX
38717
38718                                   ; ------------------------------------------------
38719                                   ; HU10--ES - MCB inside UMB; if it's a system MCB,
38720                                   ;            we're not in the same UMB, so exit.
38721                                   ; ------------------------------------------------
38722
38723 0000326D E84AFF                   hu10:       call   isSysMCB       ; Returns with ZF set if owner is SYSTEM
38724 00003270 741F                        jz      short huX       ; If it is, we've finished the UMB.
38725                                      ;call   isFreeMCB       ; Returns with ZF set if owner is 0
38726 00003272 26830E010000                or      word [es:ARENA.OWNER],0
38727 00003278 7503                        jnz     short hu20
38728
38729 0000327A E81700                      call    hideMCB
38730                                   hu20:
38731                                      ;mov    al,[es:ARENA.SIGNATURE]
38732                                      ;cmp    al,arena_signature_end  ;'Z'
38733                                      ; 19/07/2023
38734 0000327D 26803E00005A                cmp     byte [es:ARENA.SIGNATURE],'Z'
38735 00003283 740C                        jz      short huX       ; 'Z' means this was the last MCB... that's it.
38736
38737                                      ;NextMCB es,ax          ; Go on forward.
38738 00003285 8CC0                        mov     ax,es
38739                                      ;add    ax,[es:3]
38740 00003287 2603060300                  add     ax,[es:ARENA.SIZE]
38741 0000328C 40                          inc     ax
38742 0000328D 8EC0                        mov     es,ax
38743
38744 0000328F EBDC                        jmp     short hu10
38745                                   huX:
38746 00003291 07                          pop     es
38747                                      ; 01/01/2023
38748                                      ;pop    ax
38749                                      ; 02/01/2023
38750 00003292 5A                          pop     dx ; (*)
38751 00003293 C3                          retn
38752
38753                                   ; 02/01/2023
38754                                   %if 0
38755
38756                                   ; ---------------------------------------------------------------------------
38757                                   ;*** isTiny - returns with ZF set if user didn't specify /S
38758                                   ; ---------------------------------------------------------------------------
38759                                   ; ENTRY:    None
38760                                   ; EXIT:     ZF set if user DIDN'T specify /S
38761                                   ; ERROR:    None
38762                                   ; USES:     Flags
38763                                   ; ---------------------------------------------------------------------------
38764
38765                                       ; 02/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
38766                                   isTiny:
38767                                       ; 02/01/2023
38768                                       ;push   ax
38769
38770                                       ;getdata al,fUmbTiny
38771                                       ;
38772                                       ;push   ds
38773                                       ;push   cs
38774                                       ;pop    ds
38775                                       ;mov    al,[fUmbTiny]
38776                                       ;pop    ds
38777
```

```
                                    ; 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:
00003294 26C70601000800             mov     word [es:ARENA.OWNER],SystemPSPOwner ; 8
0000329B 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

00003348 C3                         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
0000334A 51                         push    cx
0000334B 06                         push    es

0000334C 89C3                       mov     bx,ax                   ; Move things around... and
                                    ; 02/01/2023
                                    ;mov    ax,es                   ; save this one for later.

0000334E 268B0E0300                 mov     cx,[es:ARENA.SIZE]
                                    ; 02/01/2023
00003353 89C8                       mov     ax,cx

00003355 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
00003358 39D8                       cmp     ax,bx
0000335A 7228                       jb      short smE ; (*)

0000335C 268A160000                 mov     dl,[es:ARENA.SIGNATURE]

                                    ;mov    cx,[es:ARENA.SIZE]
                                    ; 02/01/2023
00003361 8CC0                       mov     ax,es

00003363 26891E0300                 mov     [es:ARENA.SIZE],bx
00003368 26C60600004D               mov     byte [es:ARENA.SIGNATURE],'M'

0000336E 01D8                       add     ax,bx
00003370 40                         inc     ax
00003371 8EC0                       mov     es,ax                   ; Move to new arena area

00003373 89C8                       mov     ax,cx
00003375 29D8                       sub     ax,bx
                                    ; 12/12/2022
                                    ; ax > 0
00003377 48                         dec     ax                      ; And prepare the new size

                                    ; 18/12/2022
00003378 2688160000                 mov     [es:ARENA.SIGNATURE],dl
                                    ;mov    word [es:ARENA.OWNER],0 ; (**)
0000337D 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
00003381 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>
00003384 07                         pop     es
00003385 59                         pop     cx
                                    ; 02/01/2023
                                    ;pop    bx
00003386 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:
```

```
                                    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]
00003473 2603060300                 add     ax,[es:ARENA.SIZE]
00003478 40                         inc     ax
                                    ; 22/07/2023
                                    ;mov    es,ax ; *
00003479 EBD9                       jmp     short fum10

0000347B 07                     fumX:       pop     es
                                    ; 01/01/2023
                                    ;pop    dx
                                    ;pop    cx
                                    ;pop    ax
0000347C C3                         retn

                                ; ----------------------------------------------------------------------------
                                ;*** UmbTest - returns with carry set if UMBs are not available, else CF==false
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:    None
                                ; EXIT:     Carry is clear if UMBs are available, or set if they are not
                                ; ERROR:    None
                                ; USES:     CF (AX,BX,DS,ES pushed 'cause they're used by others)
                                ; ----------------------------------------------------------------------------

                                    ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                UmbTest:
                                    ; 01/01/2023
                                    ;push   ax
0000347D 53                         push    bx ; *
                                    ;push   ds
0000347E 06                         push    es ; **

                                    ; 01/01/2023
                                    ; ds = cs

0000347F E871FB                     call    fm_link             ; Link in UMBs (if not already linked)
00003482 E80800                     call    WalkMem             ; Check to see if they're really linked
00003485 9C                         pushf                       ; And remember what we found out
00003486 E87BFB                     call    fm_unlink           ; Unlink UMBs (if WE have linked 'em)
00003489 9D                         popf                        ; And restore what we found out.

0000348A 07                         pop     es ; **
                                    ; 01/01/2023
                                    ;pop    ds
0000348B 5B                         pop     bx ; *
                                    ;pop    ax
0000348C C3                         retn

                                ; ----------------------------------------------------------------------------
                                ;*** WalkMem - travels memory chain and returns carry clear iff UMBs are linked
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:    None
                                ; EXIT:     Carry SET if MCB chain stops before 9FFF, CLEAR if stops >= 9FFF.
                                ; ERROR:    None
                                ; USES:     Flags
                                ; ----------------------------------------------------------------------------

                                    ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:3541h)

                                WalkMem:
                                    ;push   ax ; ?
                                    ;push   bx ; ?
                                    ;;push  ds ; ? ; 01/01/2023 (MSDOS 6.21 IO.SYS, SYSINIT:352Fh)
                                    ;push   es ; ? no need to save contents of these registers ?

0000348D B452                       mov     ah,GET_IN_VARS      ; Call int 21h, function 52h...
0000348F CD21                       int     21h

00003491 268B47FE                   mov     ax,[es:bx-2]
                                    ; 22/07/2023
                                um10:
00003495 8EC0                       mov     es,ax ; * ; **

                                ; ------------------------------
                                ; UM10: ES = Current MCB pointer
                                ; ------------------------------

                                ;um10:
                                    ;mov    al,[es:ARENA.SIGNATURE]
                                    ;cmp    al,arena_signature_end ; 'Z'
                                    ; 22/07/2023
00003497 26803E00005A               cmp     byte [es:ARENA.SIGNATURE],arena_signature_end ; 'Z'
0000349D 7408                       je      short um20          ; If signature == 'Z', hay no more.

                                    ;NextMCB es,bx              ; Move to the next MCB

                                    ;mov    bx,es
                                    ;;add   bx,[es:3]
                                    ;add    bx,[es:ARENA.SIZE]
                                    ;inc    bx
                                    ;mov    es,bx
                                    ; 22/07/2023
                                    ; ax = es
                                    ;mov    ax,es ; *
0000349F 2603060300                 add     ax,[es:ARENA.SIZE]
000034A4 40                         inc     ax
                                    ;mov    es,ax ; **

000034A5 EBEE                       jmp     short um10          ; And restart the loop.
                                um20:
                                    ; 22/07/2023
                                    ; ax = es
                                    ;mov    ax,es

000034A7 3DFF9F                     cmp     ax,9FFFh            ; This sets CF if ax < 9FFF.

                                    ;pop    es ; ?
                                    ;;pop   ds ; ? ; 01/01/2023 (MSDOS 6.21 IO.SYS, SYSINIT:353Dh)
                                    ;pop    bx ; ?
                                    ;pop    ax ; ?

000034AA C3                         retn

                                ; ----------------------------------------------------------------------------
                                ;*** hl_unlink - unlinks UMBs if fm_umb is set to 0; restores strategy too
                                ; ----------------------------------------------------------------------------
                                ; ENTRY:    fm_umb == 1 : leave linked, else unlink
                                ; EXIT:     None
                                ; ERROR:    None
                                ; USES:     AX, BX
                                ; ----------------------------------------------------------------------------

                                    ; 01/01/2023 - Retro DOS v4.2
                                hl_unlink:
```

```
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
```

```
39646                                   ;; ---------------------------------------------------------------------------
39647                                   ;
39648                                   ;he_unlink:
39649                                   ;   xor     bh, bh
39650                                   ;
39651                                   ;   ;getdata bl, fm_umb   ; Restore original link-state
39652                                   ;   mov     bl,[cs:fm_umb]
39653                                   ;
39654                                   ;   mov     ax,DOS_SET_UMBLINK ; 5803h
39655                                   ;   int     21h
39656                                   ;   retn
39657                                   ;
39658                                   ;%endif
39659
39660                                   ; ---------------------------------------------------------------------------
39661                                   ;*** freeUMBs - frees all HIDDEN memory elements in upper-memory.
39662                                   ; ---------------------------------------------------------------------------
39663                                   ; ENTRY:    None
39664                                   ; EXIT:     None; HIDDEN memory elements returned to FREE
39665                                   ; ERROR:    None (ignore CF)
39666                                   ; USES:     Flags
39667                                   ; ---------------------------------------------------------------------------
39668
39669                                   FreeUMBs:
39670 000034DA 50                         push    ax
39671 000034DB 06                         push    es
39672
39673 000034DC E86700                     call    HeadUmb         ; Returns with carry if err, else ES == MCB
39674 000034DF 721C                       jc      short fusX
39675                                   fus10:
39676 000034E1 8EC0                       mov     es,ax           ; Prepare for the loop; ES = current MCB addr.
39677                                   ;fus10:
39678 000034E3 E81A00                     call    isHideMCB       ; Returns with ZF set if owner is 0
39679 000034E6 7505                       jnz     short fus20
39680 000034E8 E84000                     call    freeMCB
39681                                     ; 09/09/2023
39682                                     ; ax <> es
39683 000034EB 8CC0                       mov     ax,es
39684                                   fus20:
39685                                     ;mov    al,[es:ARENA.SIGNATURE]
39686                                     ;cmp    al,arena_signature_end ; 'Z'
39687                                     ; 22/07/2023
39688 000034ED 26803E00005A               cmp     byte [es:ARENA.SIGNATURE],arena_signature_end ; 'Z'
39689 000034F3 7408                       jz      short fusX      ; That means this was the last MCB--that's it.
39690
39691                                     ; 22/07/2023
39692                                     ; ax = es
39693                                     ;mov    ax,es
39694 000034F5 2603060300                 add     ax,[es:ARENA.SIZE]
39695 000034FA 40                         inc     ax
39696                                     ; 22/07/2023
39697                                     ;mov    es,ax
39698 000034FB EBE4                       jmp     short fus10     ; Go on forward.
39699                                   fusX:
39700 000034FD 07                         pop     es
39701 000034FE 58                         pop     ax
39702 000034FF C3                         retn
39703
39704                                   ; ---------------------------------------------------------------------------
39705                                   ;*** isHideMCB - returns with ZF set if current MCB (ES:0) is HIDDEN
39706                                   ; ---------------------------------------------------------------------------
39707                                   ; ENTRY:    ES:0 should point to an MCB
39708                                   ; EXIT:     ZF set if MCB is hidden, else !ZF
39709                                   ; ERROR:    None
39710                                   ; USES:     Flags
39711                                   ; ---------------------------------------------------------------------------
39712
39713                                   isHideMCB:
39714                                       ;push   ax
39715
39716 00003500 26833E010008                cmp     word [es:ARENA.OWNER],SystemPSPOwner ; If the owner's SYSTEM
39717 00003506 7522                       jne     short ihm_x                          ; then check for HIDDEN
39718
39719                                     ;mov    ax,[es:ARENA.NAME]
39720                                     ;cmp    ax,'HI' ; 4948h
39721 00003508 26813E08004849             cmp     word [es:ARENA.NAME+0],'HI'
39722 0000350F 7519                       jne     short ihm_x
39723                                     ;mov    ax,[es:ARENA.NAME+2]
39724                                     ;cmp    ax,'DD' ; 4444h
39725 00003511 26813E0A004444             cmp     word [es:ARENA.NAME+2],'DD'
39726 00003518 7510                       jne     short ihm_x
39727                                     ;mov    ax,[es:ARENA.NAME+4]
39728                                     ;cmp    ax,'EN' ; 4E45h
39729 0000351A 26813E0C00454E             cmp     word [es:ARENA.NAME+4],'EN'
39730 00003521 7507                       jne     short ihm_x
39731                                     ;mov    ax,[es:ARENA.NAME+6]
39732                                     ;cmp    ax,'  ' ; 2020h
39733 00003523 26813E0E002020             cmp     word [es:ARENA.NAME+6],'  '
39734                                   ihm_x:
39735                                       ;pop    ax
39736 0000352A C3                         retn
39737
39738                                   ; ---------------------------------------------------------------------------
39739                                   ;*** freeMCB - marks as free the MCB at ES:0
39740                                   ; ---------------------------------------------------------------------------
39741                                   ; ENTRY:    ES:0 should point to an MCB
39742                                   ; EXIT:     None; MCB free'd
39743                                   ; ERROR:    None
39744                                   ; USES:     AX
39745                                   ; ---------------------------------------------------------------------------
39746
39747                                   freeMCB:
39748 0000352B 26C70601000000             mov     word [es:ARENA.OWNER],0
39749 00003532 B82020                     mov     ax,'  ' ; mov ax,2020h ; 31/12/2022
39750 00003535 26A30800                   mov     [es:ARENA.NAME+0],ax
39751 00003539 26A30A00                   mov     [es:ARENA.NAME+2],ax
39752 0000353D 26A30C00                   mov     [es:ARENA.NAME+4],ax
39753 00003541 26A30E00                   mov     [es:ARENA.NAME+6],ax
39754 00003545 C3                         retn
39755
39756                                   ; ---------------------------------------------------------------------------
39757                                   ;*** HeadUmb - returns in AX the address of the first UMB block (0x9FFF)
39758                                   ; ---------------------------------------------------------------------------
39759                                   ; ENTRY:  Nothing
39760                                   ; EXIT:   AX contains 0x9FFF for most systems
39761                                   ; ERROR:  Carry set if pointer is 0xFFFF (if not set up yet--DH runs into this)
39762                                   ; USES:   Flags, AX
39763                                   ; ---------------------------------------------------------------------------
39764
39765                                   HeadUmb:
39766                                       ; 13/05/2019
39767
39768                                       ;push   si ; ?
39769                                       ;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

00003558 08C0                       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
```

```
39894
39895                                   ; 01/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
39896                                   ; (SYSINIT:2B67h)
39897                                InitForConv:
39898                                   ; 11/12/2022
39899                                   ; ds = cs
39900 00003586 E83700                  call    SpaceInUMB              ; Do we have space left in the
39901                                                                   ;  current UMB ?
39902 00003589 7308                    jnc     short InitForHi                 ; yes, we have
39903 0000358B E84104                  call    ShrinkUMB               ; shrink the current UMB in use
39904 0000358E E84500                  call    GetUMBForDev            ; else try to allocate new UMB
39905                                idl0: ; 06/07/2023
39906 00003591 720D                    jc      short InitForLo                 ; we didn't succeed, so load
39907                                                                   ;  in low memory
39908                                InitForHi:
39909                                   ; 11/12/2022
39910                                   ; ds = cs
39911                                   ;mov    ax,[cs:DevUMBFree]      ; get Para addr of free mem
39912                                   ;mov    dx,[cs:DevUMBAddr]      ; UMB start addr
39913                                   ;add    dx,[cs:DevUMBSize]      ; DX = UMB End addr
39914 00003593 A1[4724]                mov     ax,[DevUMBFree]
39915 00003596 8B16[4324]              mov     dx,[DevUMBAddr]
39916 0000359A 0316[4524]              add     dx,[DevUMBSize]
39917 0000359E EB0C                    jmp     short idl1
39918
39919                                InitForLo:
39920                                   ; 11/12/2022
39921                                   ; ds = cs
39922                                   ;mov    byte [cs:DeviceHi],0    ; in case we failed to load
39923 000035A0 C606[5124]00            mov     byte [DeviceHi],0
39924                                initforlo_x:
39925                                   ; 11/12/2022
39926                                   ; ds = cs
39927                                                                   ;  into UMB indicate that
39928                                                                   ;  we are loading low
39929                                   ;mov    ax,[cs:memhi]           ; AX = start of Low memory
39930                                   ;mov    dx,[cs:ALLOCLIM]        ; DX = End of Low memory
39931 000035A5 A1[6403]                mov     ax,[memhi]
39932 000035A8 8B16[A502]              mov     dx,[ALLOCLIM]
39933                                idl1:
39934 000035AC E86600                  call    DevSetMark              ; setup a sub-arena for DD
39935                                   ; 11/12/2022
39936                                   ; ds = cs
39937                                   ;mov    [cs:DevLoadAddr],ax     ; init the Device load address
39938                                   ;mov    [cs:DevLoadEnd],dx      ; init the limit of the block
39939                                   ;mov    word [cs:DevEntry],0    ; init Entry point to DD
39940                                   ;mov    [cs:DevEntry+2],ax
39941 000035AF A3[3524]                mov     [DevLoadAddr],ax
39942 000035B2 8916[3724]              mov     [DevLoadEnd],dx
39943 000035B6 C706[3924]0000          mov     word [DevEntry],0
39944 000035BC A3[3B24]                mov     [DevEntry+2],ax
39945                                   ; 01/01/2023
39946                                   ;pop    es ; *
39947 000035BF C3                      retn
39948
39949                                ;--------------------------------------------------------------------------
39950                                ;
39951                                ; procedure : SpaceInUMB?
39952                                ;
39953                                ;   Input : DevUMBAddr, DevUMBSize, DevUMBFree & DevSize
39954                                ;   Output : Carry set if no space in UMB
39955                                ;            Carry clear if Space is available for the device in
39956                                ;              current UMB
39957                                ;
39958                                ;--------------------------------------------------------------------------
39959
39960                                SpaceInUMB:
39961                                   ; 11/12/2022
39962                                   ; ds = cs
39963                                   ;mov    ax,[cs:DevUMBSize]
39964                                   ;add    ax,[cs:DevUMBAddr]      ; End of UMB
39965                                   ;sub    ax,[cs:DevUMBFree]      ; - Free = Remaining space
39966 000035C0 A1[4524]                mov     ax,[DevUMBSize]
39967 000035C3 0306[4324]              add     ax,[DevUMBAddr]                 ; End of UMB
39968 000035C7 2B06[4724]              sub     ax,[DevUMBFree]                 ; - Free = Remaining space
39969                                   ; 11/12/2022
39970                                   ;or     ax,ax                   ; Nospace ?
39971                                   ;jnz    short spcinumb1
39972                                   ;stc
39973                                   ;retn
39974                                   ; 11/12/2022
39975 000035CB 83F801                  cmp     ax,1
39976 000035CE 7205                    jb      short spcinumb2         ; cf=1
39977                                spcinumb1:
39978 000035D0 48                      dec     ax                      ; space for sub-arena
39979                                   ; 11/12/2022
39980                                   ; ds = cs
39981 000035D1 3B06[3324]              cmp     ax,[DevSize]
39982                                   ;cmp    ax,[cs:DevSize]                 ; do we have space ?
39983                                spcinumb2:
39984 000035D5 C3                      retn
39985
39986                                ;--------------------------------------------------------------------------
39987                                ;
39988                                ; procedure : PrepareMark
39989                                ;
39990                                ;   Input : AX==Address of MCB (not addr of free space), BX==Size
39991                                ;   Output : None; MCB marked appropriately and DevUMB* set as needed.
39992                                ;
39993                                ;--------------------------------------------------------------------------
39994
39995                                ;   ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
39996                                ;
39997                                ;PrepareMark:
39998                                ;   push    ds
39999                                ;   mov     ds,ax
40000                                ;   mov     word [ARENA.OWNER],8
40001                                ;   mov     word [ARENA.NAME],'SD' ; 4453h
40002                                ;   pop     ds
40003                                ;
40004                                ;   inc     ax
40005                                ;   mov     [cs:DevUMBAddr],ax
40006                                ;   mov     [cs:DevUMBFree],ax
40007                                ;   mov     [cs:DevUMBSize],bx      ; update the UMB Variables
40008                                ;   retn
40009
40010                                ;--------------------------------------------------------------------------
40011                                ;
40012                                ; procedure : GetUMBForDev
40013                                ;
40014                                ;   Input : DevSize
40015                                ;   Output : Carry set if couldn't allocate a UMB to fit the
40016                                ;            the device.
40017                                ;            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
```

```
40142                                   ;-------------- Copy file name
40143
40144 00003625 50                           push    ax                      ; save load addr
40145
40146                                       ; 09/09/2023
40147                                       ; ds = cs
40148                                       ;lds    si,[cs:bpb_addr]        ; command line is still there
40149 00003626 C536[8103]                   lds     si,[bpb_addr]
40150
40151 0000362A 89F7                         mov     di,si
40152 0000362C FC                           cld
40153                                   dsm_again:
40154 0000362D AC                           lodsb
40155 0000362E 3C3A                         cmp     al,':'
40156 00003630 7504                         jne     short isit_slash
40157 00003632 89F7                         mov     di,si
40158 00003634 EBF7                         jmp     short dsm_again
40159                                   isit_slash:
40160 00003636 3C5C                         cmp     al, '\'
40161 00003638 7504                         jne     short isit_null
40162 0000363A 89F7                         mov     di,si
40163 0000363C EBEF                         jmp     short dsm_again
40164                                   isit_null:
40165 0000363E 08C0                         or      al,al
40166 00003640 75EB                         jnz     short dsm_again
40167 00003642 89FE                         mov     si,di
40168
40169 00003644 BF0800                       mov     di,devmark.filename ; 8
40170 00003647 B90800                       mov     cx,8                    ; maximum 8 characters
40171                                   dsm_next_char:
40172 0000364A AC                           lodsb
40173 0000364B 08C0                         or      al, al
40174 0000364D 7407                         jz      short blankout
40175 0000364F 3C2E                         cmp     al, '.'
40176 00003651 7403                         je      short blankout
40177 00003653 AA                           stosb
40178 00003654 E2F4                         loop    dsm_next_char
40179                                   blankout:
40180 00003656 E304                         jcxz    dsm_exit
40181 00003658 B020                         mov     al, ' '
40182 0000365A F3AA                         rep     stosb                   ; blank out the rest
40183                                   dsm_exit:
40184 0000365C 58                           pop     ax                      ; restore load addr
40185 0000365D 5E                           pop     si
40186 0000365E 1F                           pop     ds
40187                                       ; 03/01/2023
40188                                       ;pop    di
40189 0000365F 07                           pop     es
40190 00003660 C3                           retn
40191
40192                                   ;--------------------------------------------------------------------------
40193                                   ;
40194                                   ; procedure : SizeDevice
40195                                   ;
40196                                   ;   Input : ES:SI - points to device file to be sized
40197                                   ;
40198                                   ;   Output : Carry set if file cannot be opened or if it is an OS2EXE file
40199                                   ;
40200                                   ;   Calculates the size of the device file in paras and stores it
40201                                   ;   in DevSize
40202                                   ;
40203                                   ;--------------------------------------------------------------------------
40204
40205                                       ; 01/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
40206                                   SizeDevice:
40207                                       ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
40208                                       ; 11/12/2022 ; *
40209 00003661 1E                           push    ds ; *
40210 00003662 06                           push    es
40211 00003663 1F                           pop     ds
40212 00003664 89F2                         mov     dx,si                   ; ds:dx -> file name
40213 00003666 B8003D                       mov     ax,3D00h                ; open
40214 00003669 CD21                         int     21h
40215 0000366B 7237                         jc      short sd_err            ; open failed
40216
40217 0000366D 89C3                         mov     bx,ax                   ; BX - file handle
40218 0000366F B80242                       mov     ax,4202h                ; seek
40219 00003672 31C9                         xor     cx,cx
40220 00003674 89CA                         mov     dx,cx                   ; to end of file
40221 00003676 CD21                         int     21h
40222 00003678 7223                         jc      short sd_close          ; did seek fail (impossible)
40223 0000367A 83C00F                       add     ax,15                   ; para convert
40224 0000367D 83D200                       adc     dx,0
40225 00003680 F7C2F0FF                     test    dx,0FFF0h               ; size > 0ffffh paras ?
40226                                       ;jz     short szdev1            ; no
40227                                       ; 22/07/2023
40228 00003684 7409                         jz      short sd_ctp
40229 00003686 2EC706[3324]FFFF             mov     word [cs:DevSize],0FFFFh ; invalid device size
40230                                                                       ; assuming that we fail later
40231 0000368D EB0E                         jmp     short sd_close
40232                                   sd_ctp:
40233                                       ; 22/07/2023
40234                                   ;szdev1:
40235 0000368F B104                         mov     cl,4                    ; convert it to paras
40236 00003691 D3E8                         shr     ax,cl
40237 00003693 B10C                         mov     cl,12
40238 00003695 D3E2                         shl     dx,cl
40239 00003697 09D0                         or      ax,dx ; * ; cf=0
40240                                       ;
40241                                       ; 22/07/2023 - Retro DOS v4.2 IO.SYS (optimized)
40242                                       ; MSDOS 6.21 IO.SYS - SYSINIT:37A6h
40243                                       ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
40244                                       ;cmp    ax,[cs:DevSizeOption]
40245                                       ;ja     short szdev2
40246                                       ;mov    ax,[cs:DevSizeOption]
40247                                       ; 12/12/2022
40248                                       ;clc
40249                                   ;szdev2:
40250 00003699 2EA3[3324]                   mov     [cs:DevSize],ax             ; save file size (in paragraps)
40251                                       ; 22/07/2023
40252                                       ;clc ; cf=0 ; *       ; CLC is not needed here
40253                                                   ; (OR instruction clears CF) - E.TAN 22/07/2023
40254
40255                                       ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
40256                                       ; 12/12/2022
40257                                       ; cf=0
40258                                       ;clc
40259                                   sd_close:
40260 0000369D 9C                           pushf                           ; let close not spoil our
40261                                                                       ;  carry flag
40262 0000369E B8003E                       mov     ax,3E00h                ; close
40263 000036A1 CD21                         int     21h                     ; we are not checking for err
40264 000036A3 9D                           popf
40265                                   sd_err:
```

```
                                    ; 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]
```

```
40390 000036E9 E829DC                   call    ParaRound
40391 000036EC 2E0106[3F24]             add     [cs:DevBrkAddr+2],ax
40392 000036F1 2EC706[3D24]0000         mov     word [cs:DevBrkAddr],0
40393 000036F8 2EA1[3724]               mov     ax,[cs:DevLoadEnd]
40394 000036FC 2E3906[3F24]             cmp     [cs:DevBrkAddr+2],ax
40395 00003701 76E1                     jbe     short rba_ok
40396 00003703 E92911                   jmp     mem_err
40397                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
40398                                   ; 11/12/2022
40399                                 ;rba_ok:
40400                                 ;   retn
40401
40402                                 ;--------------------------------------------------------------------------
40403                                 ;
40404                                 ; procedure : DevSetBreak
40405                                 ;
40406                                 ;   Input : DevBrkAddr
40407                                 ;   Output : Carry set if Device returned Init failed
40408                                 ;            Else carry clear
40409                                 ;
40410                                 ;--------------------------------------------------------------------------
40411
40412                                 DevSetBreak:
40413 00003706 50                       push    ax
40414
40415 00003707 2EA1[3F24]                mov     ax,[cs:DevBrkAddr+2]   ;remove the init code
40416 0000370B 2E803E[6619]00            cmp     byte [cs:multdeviceflag],0
40417 00003711 750F                      jne     short set_break_continue ;do not check it.
40418 00003713 2E3B06[3524]              cmp     ax,[cs:DevLoadAddr]
40419 00003718 7508                      jne     short set_break_continue ;if not same, then o.k.
40420
40421                                    ;cmp    word [cs:DevBrkAddr],0
40422                                    ;je     short break_failed    ;[DevBrkAddr+2]=[memhi] & [DevBrkAddr]=0
40423                                    ; 12/12/2022
40424 0000371A 2E833E[3D24]01            cmp     word [cs:DevBrkAddr],1
40425 00003720 7204                      jb      short break_failed
40426
40427                                 set_break_continue:
40428 00003722 E8C0FF                    call    RoundBreakAddr
40429                                    ; 12/12/2022
40430 00003725 F8                        clc
40431                                 break_failed:
40432 00003726 58                        pop     ax
40433                                    ;clc
40434 00003727 C3                        retn
40435
40436                                    ; 12/12/2022
40437                                 ;break_failed:
40438                                    ;pop    ax
40439                                    ;stc
40440                                    ;retn
40441
40442                                 ;--------------------------------------------------------------------------
40443                                 ;
40444                                 ; procedure : DevBreak
40445                                 ;
40446                                 ;   Input : DevLoadAddr & DevBrkAddr
40447                                 ;   Output : none
40448                                 ;
40449                                 ;   Marks a succesful install of a device driver
40450                                 ;   Sets device size field in sub-arena &
40451                                 ;   Updates Free ptr in UMB or adjusts memhi
40452                                 ;
40453                                 ;--------------------------------------------------------------------------
40454
40455                                    ; 11/12/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
40456                                 DevBreak:
40457                                    ;push  ds ; 11/12/2022
40458
40459                                    ; 11/12/2022
40460 00003728 0E                        push    cs
40461 00003729 1F                        pop     ds
40462                                    ;mov    ax,[cs:DevLoadAddr]
40463                                    ;mov    bx,[cs:DevBrkAddr+2]
40464 0000372A A1[3524]                  mov     ax,[DevLoadAddr]
40465 0000372D 8B1E[3F24]                mov     bx,[DevBrkAddr+2]
40466                                    ; 11/12/2022
40467 00003731 1E                        push    ds
40468
40469 00003732 48                        dec     ax                      ; seg of sub-arena
40470 00003733 8ED8                      mov     ds,ax
40471 00003735 40                        inc     ax                      ; Back to Device segment
40472 00003736 29D8                      sub     ax,bx
40473 00003738 F7D8                      neg     ax                      ; size of device in paras
40474 0000373A A30300                    mov     [devmark.size],ax       ; store it in sub-arena
40475
40476                                    ; 11/12/2022
40477 0000373D 1F                        pop     ds
40478                                    ; ds = cs
40479
40480 0000373E 803E[5124]00              cmp     byte [DeviceHi],0
40481                                    ;cmp    byte [cs:DeviceHi],0
40482 00003743 7405                      je      short db_lo
40483                                    ;mov    [cs:DevUMBFree],bx      ; update Free ptr in UMB
40484                                    ;jmp    short db_exit
40485                                    ; 11/12/2022
40486 00003745 891E[4724]                mov     [DevUMBFree],bx
40487 00003749 C3                        retn
40488                                 db_lo:
40489                                    ; 11/12/2022
40490                                    ; ds = cs
40491                                    ;mov    [cs:memhi],bx
40492                                    ;mov    word [cs:memlo],0
40493 0000374A 891E[6403]                mov     [memhi],bx
40494 0000374E C706[6203]0000            mov     word [memlo],0 ; 18/12/2022
40495                                 db_exit:
40496                                    ;pop    ds ; 11/12/2022
40497                                 sd_ret:            ; 09/09/2023
40498 00003754 C3                        retn
40499
40500                                 ; 10/04/2019 - Retro DOS v4.0
40501
40502                                 ;--------------------------------------------------------------------------
40503                                 ;
40504                                 ; procedure : ParseSize
40505                                 ;
40506                                 ;   Parses the command line for SIZE= command
40507                                 ;
40508                                 ;   ES:SI = command line to parsed
40509                                 ;
40510                                 ;   returns ptr to command line after SIZE= option in ES:SI
40511                                 ;   updates the DevSizeOption variable with value supplied
40512                                 ;   in SIZE=option
40513                                 ;   Returns carry if the SIZE option was invalid
```

```
                                    ;
                                    ;--------------------------------------------------------------------------
                                    
                                        ; 02/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                        ; (SYSINIT:2D5Ah)
                                    
                                        ; 09/09/2023 - Retro DOS v4.2 IO.SYS (Optimization) ((&BugFix))
                                        ; (MSDOS 6.21 IO.SYS - SYSINIT:3871h) - Retro DOS v4.2 -
                                        ; (PCDOS 7.1 IO.SYS - SYSINIT:3D6Eh)  - Retro DOS v5.0 -
                                    ParseSize:
                                        ;push   bx
                                        ;mov    bx,si
                                    
                                        ; 09/09/2023
00003755 56                             push    si ; * ; mov bx,si
                                    
                                        ; 11/12/2022
                                        ; ds = cs
                                        ;mov    word [cs:DevSizeOption],0 ; init the value
                                        ;mov    [cs:DevCmdLine],si
                                        ;mov    [cs:DevCmdLine+2],es
00003756 C706[5224]0000                 mov     word [DevSizeOption],0 ; init the value
0000375C 8936[6224]                     mov     [DevCmdLine],si
00003760 8C06[6424]                     mov     [DevCmdLine+2],es
00003764 E82400                         call    SkipDelim
00003767 26813C5349                     cmp     word [es:si],'SI' ; 4953h
0000376C 7528                           jne     short ps_no_size
0000376E 26817C025A45                   cmp     word [es:si+2],'ZE' ; 455Ah
00003774 7520                           jne     short ps_no_size
00003776 268A4404                       mov     al,[es:si+4]
0000377A E80D10                         call    delim
                                        ;jne    short ps_no_size
                                        ; 22/07/2023
0000377D 7518                           jne     short ps_no_size_2 ; cf=0 here
0000377F 83C605                         add     si,5
00003782 E81400                         call    GetHexNum
00003785 7210                           jc      short ps_err
                                        ; 11/12/2022
                                        ; ds = cs
                                        ;mov    [cs:DevSizeOption],ax
00003787 A3[5224]                       mov     [DevSizeOption],ax
                                    
                                        ; 09/09/2023
0000378A 58                             pop     ax  ; * (discard previous si value on top of stack)
                                    
                                    ;   call    SkipDelim ; **
                                    ;
                                    ;   ; 22/07/2023
                                    ;;ps_no_size_2:
                                    ;   ; cf = 0
                                    ;   retn
                                    
                                        ; 09/09/2023
                                        ;jmp    short SkipDelim
                                    
                                    ; 09/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
                                    %if 1
                                        ; 01/11/2022
                                    SkipDelim:
                                    sd_next_char:
0000378B 268A04                         mov     al,[es:si]
0000378E E8F90F                         call    delim
00003791 75C1                           jnz     short sd_ret ; cf=0 ; 09/09/2023
00003793 46                             inc     si
00003794 EBF5                           jmp     short sd_next_char ; 01/11/2022
                                        ; 11/12/2022
                                        ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;sd_ret:
                                        ;retn
                                    %endif
                                    
                                        ;;;call SkipDelim ; **
                                        ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                        ;mov    bx,si
                                    ps_no_size:
                                        ;mov    si,bx
                                        ;pop    bx
00003796 F8                             clc     ; cf=0
                                        ;retn
                                        ; 11/12/2022
                                    ps_err:             ; cf=1
                                    ps_no_size_2:       ; 09/09/2023 (cf=0)
                                        ; 09/09/2023
00003797 5E                             pop     si ; * ; mov si,bx
                                    ;sd_ret:    ; cf=?
00003798 C3                             retn
                                    
                                    ;ps_err:
                                        ; 02/11/2022
                                        ;pop    bx
                                        ;stc
                                        ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                        ; 11/12/2022
                                        ; cf=1
                                        ;stc
                                        ; 11/12/2022
                                    ;sd_ret:
                                        ; 22/07/2023
                                        ; 12/04/2019
                                        ;retn
                                    
                                    ; 12/04/2019 - Retro DOS v4.0
                                    
                                    ;--------------------------------------------------------------------------
                                    ;
                                    ; procedure : SkipDelim
                                    ;
                                    ;   Skips delimiters in the string pointed to by ES:SI
                                    ;   Returns ptr to first non-delimiter character in ES:SI
                                    ;
                                    ;--------------------------------------------------------------------------
                                    
                                    ; 09/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
                                    %if 0
                                        ; 01/11/2022
                                    SkipDelim:
                                    sd_next_char:
                                        mov     al,[es:si]
                                        call    delim
                                        jnz     short sd_ret
                                        inc     si
                                        jmp     short sd_next_char ; 01/11/2022
                                        ; 11/12/2022
                                        ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
```

```
                                ;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
00003927 C606[5F24]FF               mov     byte [FirstUMBLinked],0FFh ; mark that 1st UMB linked
                                    ;mov    byte [cs:FirstUMBLinked],0FFh ; mark that 1st UMB linked
                                ia_1:
                                    ; 12/12/2022
                                    ; cf=0
                                    ;clc
0000392C C3                         retn
                                iau_err:
0000392D F9                         stc
                                iau_err2:
0000392E C3                         retn

                                ;--------------------------------------------------------------------------
                                ;
                                ; Procedure Name   : umb_allocate
                                ;
                                ; Inputs           : DS = data
                                ;
                                ; Outputs          : if UMB available
                                ;                       Allocates the largest available UMB and
                                ;                       BX = segment of allocated block
                                ;                       DX = size of allocated block
                                ;                       NC
                                ;                     else
                                ;                       CY
                                ;
                                ; Uses                    : BX, DX
                                ;
                                ;--------------------------------------------------------------------------

                                umb_allocate:
                                    ; 31/12/2022
                                    ; ds = cs
0000392F 50                         push    ax
00003930 B410                       mov     ah,XMM_REQUEST_UMB ; 16
00003932 BAFFFF                     mov     dx,0FFFFh               ; try to allocate largest
                                                                    ;   possible
                                    ; 31/12/2022
00003935 FF1E[4924]                 call    far [DevXMSAddr]
                                    ;call   far [cs:DevXMSAddr]
                                                                    ; dx now contains the size of
                                                                    ; the largest UMB
00003939 09D2                       or      dx,dx
0000393B 740B                       jz      short ua_err

0000393D B410                       mov     ah,XMM_REQUEST_UMB ; 16

                                    ; 31/12/2022
0000393F FF1E[4924]                 call    far [DevXMSAddr]
                                    ;call   far [cs:DevXMSAddr]

00003943 83F801                     cmp     ax,1                    ; Q: was the reqst successful
                                    ;jne    short ua_err            ; N: error
                                    ; 27/07/2023 - Retro DOS v4.2 IO.SYS (optimization)
00003946 7601                       jna     short ua_done ; if ax=1 then cf=0, else cf=1 (ax=0)
                                ua_err:
00003948 F9                         stc

                                    ;clc
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ; 12/12/2022
                                    ; cf=0
                                    ;clc
                                ua_done:
00003949 58                         pop     ax
0000394A C3                         retn
                                    ; 27/07/2023
                                ;ua_err:
                                    ;stc
                                    ;jmp    short ua_done

                                ;--------------------------------------------------------------------------
                                ;
                                ;** get_next - Find Next item in Arena
                                ;
                                ;   ENTRY   dS - pointer to block head
                                ;   EXIT    AX,ES - pointers to next head
                                ;           'C' set if arena damaged
                                ;
                                ;--------------------------------------------------------------------------

                                    ; 01/11/2022
                                get_next:
0000394B 803E00005A                 cmp     byte [0],arena_signature_end ; 'Z'
00003950 740A                       je      short gn_err
                                _get_next_:
00003952 8CD8                       mov     ax,ds                   ; ax=current block
00003954 03060300                   add     ax,[ARENA.SIZE]             ; ax=ax + current block length
00003958 40                         inc     ax                      ; remember that header!
00003959 8EC0                       mov     es,ax
                                    ;clc
                                    ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ; 11/12/2022
                                    ; cf=0
                                    ;clc
0000395B C3                         retn
                                gn_err:
0000395C F9                         stc
                                    ; 11/12/2022
                                lfu_err:    ; cf=1
0000395D C3                         retn

                                ;--------------------------------------------------------------------------
                                ;
                                ; procedure : LinkFirstUMB
                                ;
                                ;--------------------------------------------------------------------------

                                    ; 01/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                    ; (SYSINIT:2F81h)
                                LinkFirstUMB:
                                    ; 31/12/2022
                                    ; ds = cs
0000395E E8CEFF                     call    umb_allocate
00003961 72FA                       jc      short lfu_err  ; ds = cs ; 31/12/2022

                                ; bx = segment of allocated UMB
                                ; dx = size of UMB

                                    ; 31/12/2022
                                    ; ds = cs

00003963 CD12                       int     12h                     ; ax = size of memory
```

```
41134 00003965 B106                     mov     cl,6
41135 00003967 D3E0                     shl     ax,cl                   ; ax = size in paragraphs
41136
41137 00003969 89C1                     mov     cx,ax                   ; cx = size in paras
41138 0000396B 29D8                     sub     ax,bx                   ; ax = - size of unused block
41139
41140 0000396D F7D8                     neg     ax
41141
41142                                   ;sub    cx,1                    ; cx = first umb_arena
41143                                   ; 09/09/2023
41144 0000396F 49                       dec     cx
41145 00003970 8EC1                     mov     es,cx                   ; es = first umb_arena
41146
41147 00003972 26C60600004D             mov     byte [es:ARENA.SIGNATURE],arena_signature_normal ; 'M'
41148 00003978 26C70601000800           mov     word [es:ARENA.OWNER],8       ; mark as system owned
41149
41150 0000397F 26A30300                 mov     [es:ARENA.SIZE],ax
41151 00003983 26C70608005343           mov     word [es:ARENA.NAME],'SC' ; 4353h
41152
41153                                 ; put in the arena for the first UMB
41154
41155 0000398A 8EC3                     mov     es,bx                   ; es has first free umb seg
41156 0000398C 26C60600005A             mov     byte [es:ARENA.SIGNATURE],arena_signature_end ; 'Z'
41157 00003992 26C70601000000           mov     word [es:ARENA.OWNER],arena_owner_system ; 0
41158                                                                   ; mark as free
41159 00003999 4A                       dec     dx                      ; make room for arena
41160 0000399A 2689160300               mov     [es:ARENA.SIZE],dx
41161
41162                                   ;mov    es,[cs:DevDOSData]
41163                                   ; 31/12/2022
41164 0000399F 8E06[6024]               mov     es,[DevDOSData] ; ds = cs
41165 000039A3 BF8C00                   mov     di,UMB_ARENA ; 8Ch
41166 000039A6 26890D                   mov     [es:di],cx              ; initialize umb_head in DOS
41167                                                                   ;  data segment with the arena
41168                                                                   ;  just below Top of Mem
41169
41170                                 ; we must now scan the arena chain and update the size of the last arena
41171
41172 000039A9 BF2400                   mov     di,DOS_ARENA ; 24h
41173 000039AC 268E05                   mov     es,[es:di]              ; es = start arena
41174 000039AF 31FF                     xor     di,di
41175                                 ;scan_next
41176                                 ; 09/12/2022
41177                                 scannext:
41178 000039B1 26803D5A                 cmp     byte [es:di],arena_signature_end  ; 'Z'
41179 000039B5 740C                     je      short got_last
41180
41181 000039B7 8CC0                     mov     ax,es
41182 000039B9 2603060300               add     ax,[es:ARENA.SIZE]
41183 000039BE 40                       inc     ax
41184 000039BF 8EC0                     mov     es,ax
41185                                   ;jmp    short scan_next
41186                                   ; 09/12/2022
41187 000039C1 EBEE                     jmp     short scannext
41188                                 got_last:
41189                                   ;sub    word [es:ARENA.SIZE],1
41190                                   ; 09/09/2023
41191 000039C3 26FF0E0300               dec     word [es:ARENA.SIZE]
41192
41193 000039C8 26C60600004D             mov     byte [es:ARENA.SIGNATURE],arena_signature_normal ; 'M'
41194                                   ;clc
41195                                   ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
41196                                   ; 11/12/2022
41197                                   ; cf=0
41198                                   ;clc
41199 000039CE C3                       retn
41200
41201                                 ; 11/12/2022
41202                                 ;;lfu_err:
41203                                 ;   ;stc
41204                                 ;   ; 01/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
41205                                 ;   ; 11/12/2022
41206                                 ;   ; cf=1
41207                                 ;   ;stc
41208                                 ;   retn
41209
41210                                 ;--------------------------------------------------------------------------
41211                                 ;
41212                                 ; procedure : ShrinkUMB
41213                                 ;
41214                                 ;   Shrinks the current UMB in use, so that the unused portions
41215                                 ;   of the UMB is given back to the DOS free mem pool
41216                                 ;
41217                                 ;--------------------------------------------------------------------------
41218
41219                                 ShrinkUMB:
41220                                     ; 12/12/2022
41221                                     ; ds = cs
41222 000039CF 833E[4324]00             cmp     word [DevUMBAddr],0
41223                                   ;cmp    word [cs:DevUMBAddr],0
41224 000039D4 741F                     je      short su_exit
41225 000039D6 06                       push    es
41226                                   ; 01/01/2023
41227                                   ;push   bx
41228                                   ; 12/12/2022
41229                                   ;mov    bx,[cs:DevUMBFree]
41230                                   ;sub    bx,[cs:DevUMBAddr]
41231                                   ;mov    es,[cs:DevUMBAddr]
41232 000039D7 8B1E[4724]               mov     bx,[DevUMBFree]
41233 000039DB 2B1E[4324]               sub     bx,[DevUMBAddr]
41234 000039DF 8E06[4324]               mov     es,[DevUMBAddr]
41235
41236 000039E3 B8004A                   mov     ax,4A00h
41237 000039E6 CD21                     int     21h
41238                                           ; DOS - 2+ - ADJUST MEMORY BLOCK SIZE (SETBLOCK)
41239                                           ; ES = segment address of block to change
41240                                           ; BX = new size in paragraphs
41241 000039E8 8CC0                     mov     ax,es
41242 000039EA 48                       dec     ax
41243 000039EB 8EC0                     mov     es,ax
41244 000039ED 26C70601000800           mov     word [es:ARENA.OWNER],8
41245                                   ; 01/01/2023
41246                                   ;pop    bx
41247 000039F4 07                       pop     es
41248                                 su_exit:
41249 000039F5 C3                       retn
41250
41251                                 ;--------------------------------------------------------------------------
41252                                 ;
41253                                 ; procedure : UnlinkUMB
41254                                 ;
41255                                 ;   Unlinks the UMBs from the DOS arena chain
41256                                 ;
41257                                 ;--------------------------------------------------------------------------
```

```
                                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:
 00003A84 31CB              xor    bx,cx                ; remove this switch from the records
                        err_check:
 00003A86 F9                stc
                        err_chk:
                        done_swtch: ; 09/09/2023 (cf=0)
 00003A87 C3                retn
                        
                            ; 09/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
                        
                        check_switch:
 00003A88 E8240D            call   getchr
                            ;jc    short err_check
 00003A8B 72FA              jc     short err_chk
 00003A8D 24DF                 and     al,0DFh                 ; convert it to upper case
 00003A8F 3C41              cmp    al,'A'
                            ;jb    short err_check
 00003A91 72F4              jb     short err_chk ; 15/04/2019 - Retro DOS v4.0
 00003A93 3C5A              cmp    al,'Z'
 00003A95 77EF              ja     short err_check
                        
 00003A97 06                push   es
                        
 00003A98 0E                push   cs
 00003A99 07                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.

```

```
42002 00003BA9 EB24                     jmp     short got_one_secperclus_drive
42003
42004                                   ;   check for 320K
42005
42006                                   not_144m:
42007 00003BAB 833E[C24D]28             cmp     word [deviceparameters+A_DEVICEPARAMETERS.DP_CYLINDERS],40
42008 00003BB0 7521                     jne     short got_correct_mediad
42009 00003BB2 833E[D24D]08             cmp     word [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERTRACK],8
42010 00003BB7 751A                     jne     short got_correct_mediad
42011
42012 00003BB9 B3FC                     mov     bl,0FCh
42013 00003BBB EB16                     jmp     short got_correct_mediad
42014
42015                                   only_one_head:
42016
42017                                   ;   if we don't have a 360K drive, then just go use 0f0h as media descr.
42018
42019 00003BBD 803E[BF4D]00             cmp     byte [deviceparameters+A_DEVICEPARAMETERS.DP_DEVICETYPE],DEV_5INCH ; 0
42020 00003BC2 740B                     je      short got_one_secperclus_drive
42021
42022                                   ;   single sided 360K drive uses either 0fch or 0feh, depending on
42023                                   ;     whether sectorspertrack is 8 or 9. For our purposes, anything
42024                                   ;     besides 8 will be considered 0fch
42025
42026 00003BC4 B3FC                     mov     bl,0FCh                 ; single sided 9 sector media id
42027 00003BC6 833E[D24D]08             cmp     word [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERTRACK],8
42028                                   ; 12/12/2022
42029 00003BCB 7502                     jne     short got_one_secperclus_drive ; okay if anything besides 8
42030
42031 00003BCD B3FE                     mov     bl,0FEh                 ; 160K mediaid
42032
42033                                   ;   we've either got a one sided drive, or a 1.44M drive
42034                                   ;     either case we'll use 1 sector per cluster instead of 2
42035
42036                                   got_one_secperclus_drive:
42037                                       ; 03/01/2023
42038                                       ; ax = 2
42039 00003BCF 48                       dec     ax  ; ax = 1
42040 00003BD0 A2[C74D]                 mov     [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERCLUSTER],al ; 1
42041                                   ;mov    byte [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERCLUSTER],1
42042
42043                                   got_correct_mediad:
42044 00003BD3 881E[CF4D]               mov     [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.MEDIADESCRIPTOR],bl
42045
42046                                   ;    Calculate the correct number of Total Sectors on medium
42047
42048 00003BD7 A1[C24D]                 mov     ax,[deviceparameters+A_DEVICEPARAMETERS.DP_CYLINDERS]
42049 00003BDA F726[D44D]               mul     word [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.HEADS]
42050 00003BDE F726[D24D]               mul     word [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERTRACK]
42051 00003BE2 A3[CD4D]                 mov     word [deviceparameters+A_DEVICEPARAMETERS.DP_BPB+A_BPB.TOTALSECTORS],ax
42052 00003BE5 F8                       clc                             ; we currently return no errors
42053
42054 00003BE6 C3                       retn
42055
42056                                   ;   M047 -- end rewritten routine
42057
42058                                   ;--------------------------------------------------------------------------
42059                                   ;
42060                                   ; procedure : organize
42061                                   ;
42062                                   ;--------------------------------------------------------------------------
42063
42064                                   ; 09/09/2023 - Retrodos v4.2 IO.SYS (Optimization)
42065                                   %if 1
42066                                   end_commd_line:
42067 00003BE7 AA                       stosb                           ; store line feed char in buffer for the linecount.
42068                                   ;mov    byte [cs:com_level],0 ; reset the command level.
42069                                   ; 03/01/2023
42070                                   ; ds = cs
42071                                   ;mov    byte [com_level],0
42072                                   ;jmp    short org1
42073                                   ; 09/09/2023
42074 00003BE8 EB0E                     jmp     short org0
42075                                   nochar1:
42076 00003BEA F9                       stc
42077 00003BEB C3                       retn
42078                                   %endif
42079                                       ; 02/11/2022 - Retrodos v4.0 (Modified MSDOS 5.0 IO.SYS)
42080                                       ; (SYSINIT:3234h)
42081
42082                                       ; 03/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
42083                                       ; (SYSINIT:4067h)
42084
42085                                       ; 09/09/2023 - Retrodos v4.2 IO.SYS (Optimization)
42086                                       ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:42D9h)
42087
42088                                   organize:
42089                                       ; 03/01/2023
42090                                       ; ds = cs
42091 00003BEC 8B0E[5603]               mov     cx,[count]
42092                                   ;mov    cx,[cs:count]
42093 00003BF0 E3F8                     jcxz    nochar1
42094
42095                                   ;ifndef     MULTI_CONFIG
42096                                   ;
42097                                   ;;   In MULTI_CONFIG, we map to upper case on a line-by-line basis,
42098                                   ;;   because we the case of values in SET commands preserved
42099                                   ;
42100                                   ;   call    mapcase
42101                                   ;endif
42102                                       ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
42103                                       ; 03/01/2023 - Retro DOS v4.2
42104                                       ;call   mapcase
42105
42106 00003BF2 31F6                     xor     si,si
42107 00003BF4 89F7                     mov     di,si
42108 00003BF6 31C0                     xor     ax,ax
42109                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
42110                                   ;;mov   byte [cs:com_level],0
42111                                   ; 12/12/2022
42112                                   ;mov    [cs:com_level],al ; 0
42113                                   ; 03/01/2023
42114                                   ; ds = cs
42115                                   ; 09/09/2023
42116                                   ;mov    [com_level],al ; 0
42117                                   org0:
42118 00003BF8 C606[5003]00             mov     byte [com_level],0 ; 09/09/2023
42119                                   org1:
42120 00003BFD E8EF01                   call    skip_comment
42121 00003C00 74E5                     jz      short end_commd_line  ; found a comment string and skipped.
42122 00003C02 E8D001                   call    get2                  ; not a comment string. then get a char.
42123 00003C05 3C0A                     cmp     al,lf ; 0Ah
42124 00003C07 74DE                     je      short end_commd_line  ; starts with a blank line.
42125 00003C09 3C20                     cmp     al,' ' ; 20h
```

```
                            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
```

```
42250 00003C5D 75B9                     jnz     short findcom
42251                                 gotcom0:
42252 00003C5F 5F                       pop     di
42253 00003C60 5E                       pop     si
42254 00003C61 59                       pop     cx
42255 00003C62 EB10                     jmp     short gotcom
42256                                 nocom:
42257 00003C64 5F                       pop     di
42258 00003C65 5E                       pop     si
42259 00003C66 59                       pop     cx
42260 00003C67 B05A                         mov     al,CONFIG_UNKNOWN  ; 'Z'
42261 00003C69 AA                       stosb                           ; save indicator char.
42262                                 _skipline:
42263 00003C6A E86801                   call    get2
42264 00003C6D 3C0A                     cmp     al,lf ; 0Ah             ; skip this bad command line
42265 00003C6F 75F9                         jne     short _skipline
42266                                   ;jmp    short end_commd_line  ; handle next command line
42267                                   ; 09/09/2023
42268 00003C71 E973FF                   jmp     end_commd_line
42269                                 gotcom:
42270 00003C74 AA                           stosb                               ; save indicator char in buffer
42271
42272                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
42273
42274                                 ; 03/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
42275
42276                                 ;ifdef      MULTI_CONFIG
42277
42278                                 ;   Don't pollute "cmd_indicator" with the CONFIG_OPTION_QUERY bit though;
42279                                 ;   it screws up the direct comparisons below.
42280
42281 00003C75 247F                     and     al,~CONFIG_OPTION_QUERY ; 7Fh
42282                                 ;endif
42283                                   ;mov    [cs:cmd_indicator],al ; save it for the future use.
42284                                   ; 03/01/2023
42285                                   ; ds = cs
42286 00003C77 A2[5403]                 mov     [cmd_indicator],al    ; save it for the future use.
42287
42288                                 ;ifdef      MULTI_CONFIG
42289
42290                                 ;   There is no whitespace/delimiter between the "begin block" character
42291                                 ;   ([) and the name of block (eg, [menu]), therefore skip this delimiter
42292                                 ;   skipping code
42293
42294 00003C7A 3C5B                     cmp     al,CONFIG_BEGIN
42295 00003C7C 7455                     je      short org31
42296 00003C7E 3C4F                     cmp     al,CONFIG_SUBMENU ; 'O'
42297 00003C80 740F                     je      short no_mapcase
42298 00003C82 3C45                     cmp     al,CONFIG_MENUITEM ; 'E'
42299 00003C84 740B                     je      short no_mapcase
42300 00003C86 3C41                     cmp     al,CONFIG_MENUDEFAULT ; 'A'
42301 00003C88 7407                     je      short no_mapcase
42302 00003C8A 3C4A                     cmp     al,CONFIG_INCLUDE ; 'J'
42303 00003C8C 7403                     je      short no_mapcase
42304 00003C8E E8350B                   call    mapcase                 ; map case of rest of line to UPPER
42305                                 no_mapcase:
42306                                 ;endif
42307                                   ;; 02/11/2022
42308                                   ;;mov   [cs:cmd_indicator],al ; save it for the future use.
42309                                   ;; 03/01/2023
42310                                   ;; ds = cs
42311                                   ;mov    [cmd_indicator],al
42312                                 org2:
42313 00003C91 E84101                   call    get2                    ; skip the command name until delimiter
42314 00003C94 3C0A                         cmp     al,lf           ; 0Ah
42315 00003C96 740F                     je      short org21
42316 00003C98 3C0D                     cmp     al,cr       ; 0Dh
42317 00003C9A 740B                     je      short org21
42318                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
42319                                   ; 03/01/2023 - Retro DOS v4.2
42320 00003C9C 3C2F                     cmp     al, '/'                 ; T-RICHJ: Added to allow DEVHIGH/L:...
42321 00003C9E 7407                     je      short org21             ; T-RICHJ: to be parsed properly.
42322
42323 00003CA0 E8E70A                   call    delim
42324 00003CA3 75EC                         jnz short org2
42325 00003CA5 EB02                     jmp     short org3
42326                                 org21:                                  ;if cr or lf then
42327 00003CA7 4E                       dec     si                      ; undo si, cx register
42328 00003CA8 41                       inc     cx                      ;  and continue
42329                                 org3:
42330                                   ;cmp    byte [cs:cmd_indicator],CONFIG_COMMENT ; 'Y'
42331                                   ;je     short get_cmt_token
42332                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
42333                                   ; 03/01/2023 - Retro DOS v4.2
42334                                   ;cmp    byte [cs:cmd_indicator],CONFIG_DEVICE ; 'D'
42335                                   ;je     short org_file
42336                                   ;cmp    byte [cs:cmd_indicator],CONFIG_INSTALL ; 'I'
42337                                   ;je     short org_file
42338                                   ;cmp    byte [cs:cmd_indicator],CONFIG_INSTALLHIGH ; 'W'
42339                                   ;je     short org_file
42340                                   ; 02/11/2022
42341                                   ; 03/01/2023 - Retro DOS v4.2
42342                                   ;;cmp   byte [cs:cmd_indicator],CONFIG_DEVICE ; 'D'
42343                                   ;;je    short org_file
42344                                   ;cmp    byte [cs:cmd_indicator],CONFIG_SHELL ; 'S'
42345                                   ;je     short org_file
42346                                       ;cmp       byte [cs:cmd_indicator],CONFIG_SWITCHES ; '1'
42347                                   ;je     short org_switch
42348
42349                                   ; 03/01/2023
42350                                   ; ds = cs
42351
42352 00003CA9 803E[5403]59               cmp     byte [cmd_indicator],CONFIG_COMMENT ; 'Y'
42353 00003CAE 745D                     je      short get_cmt_token
42354                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
42355                                   ; 03/01/2023 - Retro DOS v4.2
42356 00003CB0 803E[5403]44             cmp     byte [cmd_indicator],CONFIG_DEVICE ; 'D'
42357 00003CB5 7430                     je      short org_file
42358 00003CB7 803E[5403]49             cmp     byte [cmd_indicator],CONFIG_INSTALL ; 'I'
42359 00003CBC 7429                     je      short org_file
42360 00003CBE 803E[5403]57             cmp     byte [cmd_indicator],CONFIG_INSTALLHIGH ; 'W'
42361 00003CC3 7422                     je      short org_file
42362                                   ; 02/11/2022
42363                                   ; 03/01/2023 - Retro DOS v4.2
42364                                   ;cmp    byte [cmd_indicator],CONFIG_DEVICE ; 'D'
42365                                   ;je     short org_file
42366 00003CC5 803E[5403]53                 cmp byte [cmd_indicator],CONFIG_SHELL ; 'S'
42367 00003CCA 741B                     je      short org_file
42368 00003CCC 803E[5403]31                 cmp byte [cmd_indicator],CONFIG_SWITCHES ; '1'
42369 00003CD1 7403                     je      short org_switch
42370
42371                                 org31:
42372 00003CD3 E99500                   jmp     org4
42373
```

```
                                org_switch:
00003CD6 E81601                     call    skip_comment
00003CD9 7472                       jz      short end_commd_line_brdg

00003CDB E8F700                     call    get2
00003CDE E8B10A                     call    org_delim
00003CE1 74F3                       jz      short org_switch

00003CE3 AA                         stosb
00003CE4 E99300                     jmp     org5

                                org_file:                   ; get the filename and put 0 at end
00003CE7 E80501                     call    skip_comment
00003CEA 7464                       jz      short org_put_zero

00003CEC E8E600                     call    get2            ; not a comment
00003CEF E8980A                     call    delim
00003CF2 74F3                       jz      short org_file ; skip the possible delimiters

00003CF4 AA                         stosb                   ; copy the first non delim char found in buffer

                                org_copy_file:
00003CF5 E8F700                     call    skip_comment    ; comment char in the filename?
00003CF8 7456                       jz      short org_put_zero ; then stop copying filename at that point

00003CFA E8D800                     call    get2
00003CFD 3C2F                       cmp     al,'/'          ; a switch char? (device=filename/xxx)
00003CFF 7457                       je      short end_file_slash ; this will be the special case.

00003D01 AA                         stosb                   ; save the char. in buffer
00003D02 E8850A                     call    delim
00003D05 7459                       jz      short end_copy_file

00003D07 3C20                       cmp     al, ' '
00003D09 77EA                       ja      short org_copy_file ; keep copying
00003D0B EB53                       jmp     short end_copy_file ; otherwise, assume end of the filename.

                                get_cmt_token:                  ; get the token. just max. 2 char.
00003D0D E8C500                     call    get2
00003D10 3C20                       cmp     al,' '          ; skip white spaces or "=" char.
00003D12 74F9                       je      short get_cmt_token ; (we are allowing the other special
00003D14 3C09                       cmp     al,tab ; 9      ;  characters can used for comment id.
00003D16 74F5                       je      short get_cmt_token ;  character.)
00003D18 3C3D                       cmp     al,'='          ; = is special in this case.
00003D1A 74F1                       je      short get_cmt_token
00003D1C 3C0D                       cmp     al,cr
00003D1E 7426                       je      short get_cmt_end ; cannot accept the carriage return
00003D20 3C0A                       cmp     al,lf
00003D22 7422                       je      short get_cmt_end

                                    ; 03/01/2023
                                    ; ds = cs
                                    ;mov    [cs:cmmt1],al  ; store it
                                    ;mov    byte [cs:cmmt],1 ; 1 char. so far.
00003D24 A2[5203]                   mov     [cmmt1],al      ; store it
00003D27 C606[5103]01               mov     byte [cmmt],1  ; 1 char. so far.
00003D2C E8A600                     call    get2
00003D2F 3C20                       cmp     al,' ' ; 20h
00003D31 7413                       je      short get_cmt_end
00003D33 3C09                       cmp     al,tab ; 9
00003D35 740F                       je      short get_cmt_end
00003D37 3C0D                       cmp     al,cr  ; 0Dh
00003D39 740B                       je      short get_cmt_end
00003D3B 3C0A                       cmp     al,lf  ; 0Ah
00003D3D 740E                       je      short end_commd_line_brdg

                                    ;mov    [cs:cmmt2],al
                                    ;inc    byte [cs:cmmt]
                                    ; 03/01/2023
00003D3F A2[5303]                   mov     [cmmt2],al
00003D42 FE06[5103]                 inc     byte [cmmt]

                                get_cmt_end:
00003D46 E88C00                     call    get2
00003D49 3C0A                       cmp     al,lf
00003D4B 75F9                       jne     short get_cmt_end     ; skip it.
                                end_commd_line_brdg:
00003D4D E997FE                     jmp     end_commd_line        ; else jmp to end_commd_line

                                org_put_zero:                           ; make the filename in front of
00003D50 26C60500                   mov     byte [es:di],0        ;  the comment string to be an asciiz.
00003D54 47                         inc     di
00003D55 E98FFE                     jmp     end_commd_line        ;  (maybe null if device=/*)

                                end_file_slash:                 ; al = "/" option char.
00003D58 26C60500                   mov     byte [es:di],0        ; make a filename an asciiz
00003D5C 47                         inc     di                    ; and
00003D5D AA                         stosb                         ; store "/" after that.
00003D5E EB1A                       jmp     short org5            ; continue with the rest of the line

                                end_copy_file:
00003D60 26C645FF00                 mov     byte [es:di-1],0      ; make it an asciiz and handle the next char.
00003D65 3C0A                       cmp     al,lf
00003D67 74E4                       je      short end_commd_line_brdg
00003D69 EB0F                       jmp     short org5

                                org4:                                   ; org4 skips all delimiters after the command name except for '/'
00003D6B E88100                     call    skip_comment
00003D6E 74DD                       jz      short end_commd_line_brdg

00003D70 E86200                     call    get2
00003D73 E81C0A                     call    org_delim             ; skip delimiters except '/' (mrw 4/88)
00003D76 74F3                       jz      short org4
00003D78 EB08                       jmp     short org51

                                org5:                                   ; rest of the line
00003D7A E87200                     call    skip_comment          ; comment?
00003D7D 74CE                       jz      short end_commd_line_brdg
00003D7F E85300                     call    get2                  ; not a comment.

                                org51:
00003D82 AA                         stosb                         ; copy the character
00003D83 3C22                       cmp     al,'"'  ; 22h         ; a quote ?
00003D85 743A                       je      short at_quote
00003D87 3C20                       cmp     al,' '  ; 20h
00003D89 77EF                       ja      short org5

                                    ; 09/09/2023
                                    ; (Note: PCDOS 7.1 IBMBIO.COM does not contain M051 modification)

                                                                    ; M051 - Start
                                    ; 03/01/2023
                                    ; ds = cs
00003D8B 803E[5403]55                  cmp byte [cmd_indicator],CONFIG_DEVICEHIGH
```

```
                                    ;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
                                
00003DF1 803E[5003]00               cmp     byte [com_level],0
                                    ;cmp    byte [cs:com_level],0 ; only check it if parameter level is 0.
00003DF6 752C                       jne     short no_commt  ; (not inside quotations)
                                
00003DF8 803E[5103]01               cmp     byte [cmmt],1
                                    ;cmp    byte [cs:cmmt],1
00003DFD 7225                       jb      short no_commt
                                
00003DFF 268A04                     mov     al,[es:si]
                                
00003E02 3806[5203]                 cmp     [cmmt1],al
                                    ;cmp    [cs:cmmt1],al
00003E06 751C                       jne     short no_commt
                                
00003E08 803E[5103]02               cmp     byte [cmmt],2
                                    ;cmp    byte [cs:cmmt],2
00003E0D 750A                       jne     short skip_cmmt
                                
00003E0F 268A4401                   mov     al,[es:si+1]
                                
00003E13 3806[5303]                 cmp     [cmmt2],al
                                    ;cmp    [cs:cmmt2],al
00003E17 750B                       jne     short no_commt
                                skip_cmmt:
00003E19 E3C0                       jcxz    noget           ; get out of organize routine.
00003E1B 268A04                     mov     al,[es:si]
00003E1E 46                         inc     si
00003E1F 49                         dec     cx
00003E20 3C0A                       cmp     al,lf           ; line feed?
00003E22 75F5                       jne     short skip_cmmt
                                no_commt:
00003E24 C3                         retn
                                
                                ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
                                ; (SYSINIT:42C8h)
                                
                                ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ;%if 0
                                
                                ;ifdef      MULTI_CONFIG
                                
                                ;--------------------------------------------------------------------------
                                ;
                                ;   kbd_read: wait for keystroke
                                ;
                                ;   INPUT
                                ;       DS == CS == sysinitseg
                                ;
                                ;   OUTPUT
                                ;       Carry SET to clean boot, CLEAR otherwise
                                ;
                                ;   OTHER REGS USED
                                ;       All
                                ;
                                ;   HISTORY
                                ;       Created 16-Nov-1992 by JeffPar
                                ;
                                ;--------------------------------------------------------------------------
                                
                                kbd_read:
00003E25 F606[814C]02               test    byte [bDisableUI],2
00003E2A 7520                       jnz     short kbd_nodelay
                                
00003E2C 1E                         push    ds              ; the bios timer tick count is incremented
00003E2D 29C0                       sub     ax,ax           ; 18.2 times per second;
00003E2F 8ED8                       mov     ds,ax           ; watch the timer tick count for 37 transitions
                                    ;mov    dx,[046Ch]     ; get initial value
                                kbd_loop:
00003E31 B401                       mov     ah,1            ;
00003E33 CD16                       int     16h             ; peek the keyboard
00003E35 7514                       jnz short kbd_loopdone ; something's there, get out
00003E37 B402                       mov     ah,2            ; peek the shift states
00003E39 CD16                       int     16h             ;
00003E3B A803                       test    al,03h          ; either right or left shift key bits set?
00003E3D 750C                       jnz short kbd_loopdone ; yes
00003E3F A16C04                     mov     ax,[046Ch];
                                    ;sub    ax,dx           ; get difference
                                    ; 15/04/2019 - Retro DOS v4.0
00003E42 2E2B06[8C03]               sub     ax,[cs:_timer_lw_] ; MSDOS 6.21 IO.SYS - SYSINIT:42E5h
                                
00003E47 3C25                       cmp     al,37           ; reached limit?  ; (2 seconds)
00003E49 72E6                       jb  short kbd_loop ; not yet
                                kbd_loopdone:
00003E4B 1F                         pop     ds              ; delay complete!
                                kbd_nodelay:
00003E4C 29DB                       sub     bx,bx           ; assume clean boot
00003E4E B402                       mov     ah,2            ; peek the shift states
00003E50 CD16                       int     16h             ;
00003E52 A803                       test    al,03h          ; either right or left shift key bits set?
00003E54 7407                       jz      short kbd_notshift ; no
00003E56 43                         inc     bx              ; yes
00003E57 43                         inc     bx
                                    ; MSDOS 6.21 IO.SYS - SYSINIT:4301h
00003E58 800E[854C]04               or      byte [bQueryOpt],4
                                kbd_notshift:                   ;
00003E5D B401                       mov     ah,1            ; peek the keyboard
00003E5F CD16                       int     16h             ;
00003E61 743E                       jz  short kbd_test ; no key present
00003E63 08C0                       or      al,al           ; is it a function key?
00003E65 753A                       jnz short kbd_test ; no
                                
                                    ; MSDOS 6.21 IO.SYS - SYSINIT:430Bh
00003E67 80FC62                     cmp     ah,62h          ; CTRL F5
00003E6A 7405                       je  short kbd_cfg_bypass
                                
00003E6C 80FC3F                     cmp     ah,3Fh          ; F5 function key?
00003E6F 750D                       jne short kbd_notf5       ; no
                                kbd_cfg_bypass:
00003E71 BA[FC51]                   mov     dx,_$CleanMsg
00003E74 E8DD0B                     call    print
                                    ; MSDOS 6.21 IO.SYS - SYSINIT:431Bh
00003E77 800E[854C]04               or      byte [bQueryOpt],4
00003E7C EB16                       jmp     short kbd_eat   ; yes, clean boot selected
                                kbd_notf5:
                                    ; MSDOS 6.21 IO.SYS - SYSINIT:4322h
00003E7E 80FC65                     cmp     ah,65h          ; CTRL F8
00003E81 7405                       je  short kbd_cfg_confirm
                                
00003E83 80FC42                     cmp     ah,42h          ; F8 function key?
00003E86 7523                       jne short kbd_exit ; no
                                kbd_cfg_confirm:
00003E88 BA[3A52]                   mov     dx,_$InterMsg
```

```
00003E8B E8C60B                         call    print           ;
00003E8E B301                           mov     bl,1            ; yes, interactive-boot option enabled
00003E90 881E[854C]                     mov     [bQueryOpt],bl  ; change default setting
                                kbd_eat:                        ;
00003E94 B400                           mov     ah,0            ;
00003E96 CD16                           int     16h             ; eat the key we assumed was a signal
00003E98 C606[8B4C]FF                   mov byte [secElapsed],-1
00003E9D 09DB                           or      bx,bx           ;
00003E9F 7405                           jz  short kbd_clean      ;
                                kbd_test:                       ;
00003EA1 80FB02                         cmp     bl,2            ;
00003EA4 7205                           jb  short kbd_exit ;
                                kbd_clean:                      ;
00003EA6 E86E08                         call    disable_autoexec; yes, tell COMMAND to skip autoexec.bat
00003EA9 F9                             stc                     ; set carry to indicate abort
00003EAA C3                             retn                     ;
                                kbd_exit:                       ;
00003EAB F8                             clc                     ; clear carry to indicate success
00003EAC C3                             retn                     ;

                                ;--------------------------------------------------------------------------
                                ;
                                ;   set_numlock: set numlock LED
                                ;
                                ;   INPUT
                                ;       ES:SI -> numlock setting (ie, "ON" or "OFF")
                                ;
                                ;   OUTPUT
                                ;       None
                                ;
                                ;   OTHER REGS USED
                                ;       None
                                ;
                                ;   HISTORY
                                ;       Created 16-Nov-1992 by JeffPar
                                ;
                                ;--------------------------------------------------------------------------
                                    ; 04/01/2023 - Retro DOS v4.2

                                set_numlock:
                                        ; 04/01/2023
                                    ;push   ax
00003EAD 1E                             push    ds
00003EAE 29C0                           sub     ax,ax
00003EB0 8ED8                           mov     ds,ax
00003EB2 268B04                         mov     ax,[es:si]      ; get 1st 2 bytes of value (ON or OF)
00003EB5 2E3B06[C451]                   cmp     ax,[cs:OnOff+2]   ; should we turn it off?
00003EBA 7507                           jne short not_off  ; no
00003EBC 80261704DF                     and     byte [0417h],~20h ; 0DFh
00003EC1 EB0D                           jmp     short set_done
                                not_off:
00003EC3 2E3B06[C251]                   cmp     ax,[cs:OnOff]     ; should we turn it on?
00003EC8 F9                             stc
00003EC9 7505                           jne short set_done ; no
00003ECB 800E170420                     or      byte [0417h],20h
                                set_done:
00003ED0 1F                             pop     ds
                                    ; 04/01/2023
                                    ;pop    ax
00003ED1 C3                             retn

                                ; 16/04/2019 - Retro DOS v4.0

                                ;--------------------------------------------------------------------------
                                ;
                                ;   menu_check:  check for presence of menu (and other) configuration blocks
                                ;
                                ;   INPUT
                                ;       CX == "organized" config.sys memory image length
                                ;    ES:SI -> "organized" config.sys memory image
                                ;       DS == CS == sysinitseg
                                ;
                                ;   OUTPUT
                                ;       Same as above; the idea is that menu_check simply transforms
                                ;       a block-structured config.sys image into a conventional image,
                                ;       based on the user's block selection and any other boot-time options
                                ;       the user may have employed...
                                ;
                                ;   OTHER REGS USED
                                ;       All
                                ;
                                ;   NOTES
                                ;       [count] and [org_count] are set to the new config.sys image length
                                ;
                                ;   HISTORY
                                ;       Created 16-Mar-1992 by JeffPar
                                ;
                                ;--------------------------------------------------------------------------

                                    ; 04/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:4378h)

                                menu_check:

                                ;   Search for SWITCHES, determine if /N or /F are present; if so, then
                                ;   disable clean/interactive boot options

00003ED2 51                             push    cx              ;
00003ED3 56                             push    si              ;
00003ED4 29DB                           sub     bx,bx           ; remains ZERO until first block
                                swchk_loop:                     ;
00003ED6 E83507                         call    get_char        ; get first char of current line
00003ED9 724C                           jc  short swchk_end       ; hit eof
00003EDB 3C5B                           cmp     al,CONFIG_BEGIN ; '['
00003EDD 7503                           jne short swchk_next1 ;
00003EDF 43                             inc     bx              ; remember that we've seen a block
00003EE0 EB40                           jmp     short swchk_nextline
                                swchk_next1:                    ;
00003EE2 3C4E                           cmp     al,CONFIG_NUMLOCK
00003EE4 750E                           jne short swchk_next2 ;
00003EE6 09DB                           or      bx,bx           ; only do NUMLOCK commands that exist
00003EE8 7538                           jnz short swchk_nextline ; before the first block
00003EEA E8C0FF                         call    set_numlock     ; REM it out so we don't act on it later, too
00003EED 26C644FF30                     mov     byte [es:si-1],CONFIG_REM
00003EF2 EB2E                           jmp     short swchk_nextline
                                swchk_next2:                    ;
00003EF4 3C31                           cmp     al,CONFIG_SWITCHES
00003EF6 752A                           jne short swchk_nextline ; this line ain't it
                                swchk_scan:                     ;
00003EF8 E81307                         call    get_char        ; look for /N or /F
                                swchk_scan1:                    ;
00003EFB 3C0A                           cmp     al,LF           ; end of line?
00003EFD 7423                           je  short swchk_nextline ; yes
```

```
42870 00003EFF 3C2F                     cmp     al,'/'          ; switch-char?
42871 00003F01 75F5                     jne short swchk_scan ; no
42872 00003F03 E80807                   call    get_char        ;
42873 00003F06 24DF                     and     al,~20h ; 0DFh    ; convert to upper case
42874 00003F08 3A06[6323]               cmp     al,[swit_n+1]     ; 'N'
42875 00003F0C 7507                     jne short swchk_scan2 ; no
42876 00003F0E 800E[814C]01             or      byte [bDisableUI],1
42877 00003F13 EBE3                     jmp short swchk_scan ; continue looking for switches of interest
42878                                 swchk_scan2:                    ;
42879 00003F15 3A06[6F23]               cmp     al,[swit_f+1]     ; 'F'
42880 00003F19 75E0                     jne short swchk_scan1 ; no
42881 00003F1B 800E[814C]02             or      byte [bDisableUI],2
42882 00003F20 EBD6                     jmp     short swchk_scan ; continue looking for switches of interest
42883                                 swchk_nextline:                 ;
42884 00003F22 E8C306                   call    skip_opt_line   ;
42885 00003F25 EBAF                     jmp     short swchk_loop ;
42886                                 swchk_end:                      ;
42887 00003F27 5E                       pop     si              ;
42888 00003F28 59                       pop     cx              ;
42889
42890                                 ;   Do the keyboard tests for clean/interactive boot now, but only if
42891                                 ;   the DisableUI flag is still clear
42892
42893 00003F29 F606[814C]01                 test    byte [bDisableUI],1
42894 00003F2E 7508                         jnz short menu_search
42895                                 ;
42896                                 ;   Wait for 2 seconds first, UNLESS the /F bit was set in bDisableUI, or
42897                                 ;   there is anything at all in the keyboard buffer
42898                                 ;
42899 00003F30 E8F2FE                       call    kbd_read
42900 00003F33 7303                         jnc short menu_search
42901 00003F35 E9EE01                       jmp menu_abort
42902
42903                                 ;   Search for MENU block; it is allowed to be anywhere in config.sys
42904
42905                                 menu_search:
42906 00003F38 29DB                     sub     bx,bx           ; if no MENU, default to zero for no_selection
42907 00003F3A BF[C64C]                 mov     di,szMenu ;
42908 00003F3D E80304                   call    find_block      ; find the MENU block
42909 00003F40 7337                     jnc short menu_found ;
42910 00003F42 C606[BE4C]00             mov     byte [szBoot],0
42911 00003F47 E90C02                   jmp no_selection ; not found
42912
42913                                 ;   Process the requested menu color(s)
42914
42915                                 menu_color:
42916 00003F4A 51                       push    cx              ;
42917 00003F4B 52                       push    dx              ;
42918                                   ;;mov   dx,0007h        ; default color setting
42919                                   ; 10/09/2023
42920                                   ;mov    dl,7 ; !*!
42921 00003F4C E89E06                   call    get_number      ; get first number
42922 00003F4F 80E30F                   and     bl,0Fh  ; !**! ; first # is foreground color (for low nibble)
42923 00003F52 88DD                     mov     ch,bl           ; save it in CH
42924                                   ; 01/08/2023 - Retro DOS v4.2 IO.SYS (optimization) by Erdogan Tan
42925                                   ; (high nibble of dl is 0)
42926                                   ;and    dl,0F0h ; !*!   ; (low nibble of dl would be zero)
42927                                   ;or     dl,bl           ; (low nibble of dl is 7) ! 14/08/2023
42928 00003F54 88DA                     mov     dl,bl   ; 14/08/2023
42929 00003F56 E83108                   call    delim           ; did we hit a delimiter
42930 00003F59 750E                     jne     short check_color ; no, all done
42931 00003F5B E88F06                   call    get_number      ; get next number
42932 00003F5E 80E30F                   and     bl,0Fh          ; second # is background color (for high nibble)
42933 00003F61 88DE                     mov     dh,bl           ; save it in DH
42934                                   ; 10/09/2023
42935                                   ;and    dl,0Fh  ; !**!  ;
42936 00003F63 B104                     mov     cl,4            ;
42937 00003F65 D2E3                     shl     bl,cl           ;
42938 00003F67 08DA                     or      dl,bl           ;
42939                                 check_color:
42940 00003F69 38F5                     cmp     ch,dh           ; are foreground/background the same?
42941 00003F6B 7503                     jne     short set_color       ; no
42942 00003F6D 80F208                   xor     dl,08h          ; yes, so modify the fgnd intensity
42943                                 set_color:
42944 00003F70 8816[7C4C]               mov     [bMenuColor],dl ;
42945 00003F74 5A                       pop     dx              ;
42946 00003F75 59                       pop     cx              ;
42947 00003F76 E9A900                   jmp     menu_nextitem
42948
42949                                 ;   Back to our regularly scheduled program (the COLOR and other goop
42950                                 ;   above is there simply to alleviate short jump problems)
42951
42952                                 menu_found:
42953 00003F79 C606[864C]01                 mov     byte [bDefBlock],1
42954                                       ;mov       word [offDefBlock],0
42955 00003F7E C606[8A4C]FF                 mov     byte [secTimeOut],-1
42956 00003F83 8026[854C]FD                 and     byte [bQueryOpt],~2 ; 0FDh
42957                                   ; 10/09/2023
42958 00003F88 29D2                     sub     dx,dx
42959 00003F8A 8916[884C]               mov     [offDefBlock],dx ; 0
42960
42961 00003F8E E85706                       call    skip_opt_line   ; skip to next line
42962                                   ; 10/09/2023
42963                                   ;sub    dx,dx           ; initialize total block count (0 => none yet)
42964
42965                                 ;   Process the menu block now
42966
42967                                 menu_process:
42968 00003F91 E87A06                       call    get_char        ; get first char of current line
42969 00003F94 722E                         jc  short to_menu_getdefault ; could happen if menu block at end (rare)
42970 00003F96 247F                         and     al,~CONFIG_OPTION_QUERY ; 7Fh
42971 00003F98 3C5B                         cmp     al,CONFIG_BEGIN ; BEGIN implies END
42972 00003F9A 7428                         je  short to_menu_getdefault
42973 00003F9C 3C4F                         cmp     al,CONFIG_SUBMENU
42974 00003F9E 744D                         je  short menu_item       ; go process sub-menu
42975 00003FA0 3C45                         cmp     al,CONFIG_MENUITEM
42976 00003FA2 7449                         je  short menu_item       ; go process menu item
42977 00003FA4 3C41                         cmp     al,CONFIG_MENUDEFAULT
42978 00003FA6 741E                         je  short menu_default ; go process menu default
42979 00003FA8 3C52                         cmp     al,CONFIG_MENUCOLOR
42980 00003FAA 749E                         je  short menu_color ; go process menu color
42981 00003FAC 3C4E                         cmp     al,CONFIG_NUMLOCK
42982 00003FAE 740F                         je  short menu_numlock ;
42983 00003FB0 3C30                         cmp     al,CONFIG_REM   ; allow remarks in menu block
42984 00003FB2 746E                         je  short menu_nextitem ;
42985 00003FB4 E8C307                       call    any_delim       ; allow blank lines and such
42986 00003FB7 7469                         je  short menu_nextitem ;
42987 00003FB9 F9                           stc                     ;
42988 00003FBA E82607                       call    print_error     ; non-MENU command!
42989 00003FBD EB63                         jmp     short menu_nextitem
42990                                 menu_numlock:
42991 00003FBF E8EBFE                       call    set_numlock
42992 00003FC2 EB5E                         jmp     short menu_nextitem
42993                                 to_menu_getdefault:
```

```
42994 00003FC4 EB62                             jmp     short menu_getdefault
42995
42996                                   ;   Save the offset of the default block name, we'll need it later
42997
42998                                   menu_default:
42999 00003FC6 8936[884C]                       mov     [offDefBlock],si; save address of default block name
43000 00003FCA 803E[8B4C]00                     cmp     byte [secElapsed],0
43001 00003FCF 751A                             jne short timeout_skip ; secElapsed is only zero for the FIRST menu,
43002 00003FD1 E8EA05                           call    skip_token      ; and for subsequent menus IF nothing was typed;
43003 00003FD4 724C                             jc  short menu_nextitem ; secElapsed becomes -1 forever as soon as
43004 00003FD6 E8FB05                           call    skip_delim      ; something is typed
43005 00003FD9 7247                             jc  short menu_nextitem ;
43006 00003FDB 89DE                             mov     si,bx           ;
43007 00003FDD E80D06                           call    get_number      ; get number (of seconds for timeout)
43008 00003FE0 80FB5A                           cmp     bl,90           ; limit it to a reasonable number
43009                                       ;jb    short timeout_ok ; (besides, 99 is the largest # my simple
43010 00003FE3 7602                             jna short timeout_ok ; 01/08/2023
43011 00003FE5 B35A                         mov     bl,90           ;  display function can handle)
43012                                   timeout_ok:                     ;
43013 00003FE7 881E[8A4C]                       mov     [secTimeOut],bl ;
43014                                   timeout_skip:
43015 00003FEB EB35                             jmp     short menu_nextitem
43016
43017                                   ;   Verify that this is a valid menu item by searching for the named block
43018
43019                                   menu_item:
43020                                       ;cmp   dl,9    ; 04/01/2023
43021 00003FED 80FA09                           cmp     dl,MAX_MULTI_CONFIG ; have we reached the max # of items yet?
43022 00003FF0 7330                             jae short menu_nextitem ;
43023 00003FF2 89F7                             mov     di,si           ; DS:DI -> block name to search for
43024 00003FF4 E83303                           call    srch_block      ;
43025 00003FF7 7406                             je  short menu_itemfound ;
43026 00003FF9 F9                               stc                     ;
43027 00003FFA E8E606                           call    print_error     ; print error and pause
43028 00003FFD EB23                             jmp     short menu_nextitem ; if not found, ignore this menu item
43029
43030                                   ;   srch_block, having succeeded, returns DI -> past the token that it
43031                                   ;   just matched, which in this case should be a descriptive string; ES:SI
43032                                   ;   and CX are unmodified
43033
43034                                   menu_itemfound:
43035 00003FFF 42                               inc     dx              ; otherwise, increment total block count
43036 00004000 89D3                             mov     bx,dx           ; and use it to index the arrays of offsets
43037 00004002 8887[8C4C]                       mov [abBlockType+bx],al
43038 00004006 01DB                             add     bx,bx           ; of recorded block names and descriptions
43039
43040                                   ;   There should be a description immediately following the block name on
43041                                   ;   MENUITEM line; failing that, we'll just use the block name as the
43042                                   ;   description...
43043
43044 00004008 89B7[964C]                       mov     [aoffBlockName+bx],si
43045 0000400C 89B7[AA4C]                       mov     [aoffBlockDesc+bx],si
43046 00004010 89DF                             mov     di,bx           ; skip_delim modifies BX, so stash it in DI
43047 00004012 E8A905                           call    skip_token      ;
43048 00004015 720B                             jc  short menu_nextitem ; hit eol/eof
43049 00004017 E8BA05                           call    skip_delim      ;
43050 0000401A 7206                             jc  short menu_nextitem ; hit eol/eof
43051 0000401C 87FB                             xchg    bx,di           ;
43052 0000401E 89BF[AA4C]                       mov     [aoffBlockDesc+bx],di
43053
43054                                   menu_nextitem:
43055 00004022 E8C305                           call    skip_opt_line   ;
43056 00004025 E969FF                           jmp     menu_process    ; go back for more lines
43057
43058                                   ;   Display menu items now, after determining which one is default
43059
43060                                   menu_getdefault:
43061 00004028 08D2                             or      dl,dl           ; where there any valid blocks at all?
43062 0000402A 7505                             jnz short menu_valid ; yes
43063 0000402C 29DB                             sub     bx,bx           ; no, so force autoselect of 0
43064 0000402E E9ED00                           jmp     menu_autoselect ; (meaning: process common blocks only)
43065                                   menu_valid:
43066 00004031 29DB                             sub     bx,bx           ;
43067 00004033 8816[874C]                       mov     [bMaxBlock],dl  ; first, record how many blocks we found
43068 00004037 8B3E[884C]                       mov     di,[offDefBlock];
43069 0000403B 09FF                             or      di,di           ; does a default block exist?
43070 0000403D 741C                             jz  short menu_nodefault ; no
43071 0000403F 43                               inc     bx              ; yes, walk name table, looking for default
43072                                   menu_chkdefault:
43073 00004040 53                               push    bx              ;
43074 00004041 01DB                             add     bx,bx           ;
43075 00004043 8BB7[964C]                       mov     si,[aoffBlockName+bx]
43076 00004047 B98000                           mov     cx,128          ; arbitrary maximum length of a name
43077 0000404A 1E                               push    ds              ;
43078 0000404B 06                               push    es              ;
43079 0000404C 1F                               pop     ds              ;
43080 0000404D E81A03                           call    comp_names      ; is this block the same as the default?
43081 00004050 1F                               pop     ds              ;
43082 00004051 5B                               pop     bx              ;
43083 00004052 7409                             je  short menu_setdefault ; yes
43084 00004054 43                               inc     bx              ;
43085 00004055 3A1E[874C]                       cmp     bl,[bMaxBlock]  ; all done searching?
43086 00004059 76E5                             jbe short menu_chkdefault ; not yet
43087                                   menu_nodefault:
43088 0000405B B301                             mov     bl,1            ; if no default, force default to #1
43089                                   menu_setdefault:
43090 0000405D 881E[864C]                       mov     [bDefBlock],bl  ; yes, this will be the initial current block
43091
43092                                   ;   If the timeout was explicitly set to 0 (or technically, anything that
43093                                   ;   failed to resolve to a number, like "NONE" or "EAT POTATOES"), then we're
43094                                   ;   supposed to skip menu display and run with the specified default block;
43095                                   ;   however, if the user hit Enter prior to boot, thereby requesting fully
43096                                   ;   INTERACTIVE boot, then we shall display the menu block anyway (though still
43097                                   ;   with no timeout)
43098
43099 00004061 803E[8A4C]00                       cmp     byte [secTimeOut],0 ; is timeout zero? (ie, assume default)
43100 00004066 750A                             jne short menu_display ; no
43101 00004068 F606[854C]01                     test    byte [bQueryOpt],1 ; yes, but was INTERACTIVE requested?
43102 0000406D 7503                             jnz short menu_display ; yes, so *don't* assume default after all
43103 0000406F E9C700                           jmp     not_topmenu     ;
43104
43105                                   ;   Reset the mode, so that we know screen is clean and cursor is home
43106
43107                                   menu_display:
43108 00004072 B40F                             mov     ah,0Fh          ; get current video mode
43109 00004074 CD10                             int     10h             ;
43110 00004076 B400                             mov     ah,00h          ; just re-select that mode
43111 00004078 CD10                             int     10h             ;
43112 0000407A 06                               push    es              ;
43113 0000407B B84000                           mov     ax,40h          ; reach down into the ROM BIOS data area
43114 0000407E 8EC0                             mov     es,ax           ; and save the current (default) video page
43115 00004080 26A14E00                         mov     ax,[es:004Eh]   ; start address and page #, in case the
43116 00004084 A3[834C]                         mov     [wCRTStart],ax  ; undocumented QUIET option was enabled
43117 00004087 26A06200                         mov     al,[es:0062h]   ;
```

```
43118 0000408B A2[824C]                     mov     [bCRTPage],al   ;
43119 0000408E A1[7D4C]                     mov     ax,[bMenuPage]    ; select new page for menu
43120 00004091 CD10                         int     10h             ;
43121 00004093 B80006                       mov     ax,0600h        ; clear entire screen
43122 00004096 8A3E[7C4C]                   mov     bh,[bMenuColor] ; using this color
43123 0000409A 29C9                         sub     cx,cx           ; upper left row/col
43124                                       ;mov       dl,[es:CRT_Cols]
43125 0000409C 268A164A00                   mov dl,[es:4Ah]
43126 000040A1 FECA                     dec     dl              ;
43127                                       ;mov       dh,[es:CRT_Rows];
43128 000040A3 268A368400                   mov dh,[es:84h]
43129 000040A8 08F6                     or      dh,dh           ; # of rows valid?
43130 000040AA 7504                         jnz short menu_clear ; hopefully
43131 000040AC 8A36[804C]                   mov     dh,[bLastRow]   ; no, use a default
43132                                 menu_clear:
43133 000040B0 CD10                         int     10h             ; clear the screen using the req. attribute
43134 000040B2 07                           pop     es              ;
43135 000040B3 8836[804C]                   mov     [bLastRow],dh   ; save DH
43136 000040B7 BA[7752]                     mov     dx,_$MenuHeader
43137 000040BA E89709                       call    print           ; cursor now on row 3 (numbered from 0)
43138
43139 000040BD F606[814C]01                 test    byte [bDisableUI],1
43140 000040C2 751F                         jnz     short menu_nostatus
43141 000040C4 8A3E[7D4C]                   mov     bh,[bMenuPage]  ;
43142 000040C8 8A36[804C]                   mov     dh,[bLastRow]   ; restore DH
43143 000040CC B200                         mov     dl,0            ; print the status line on row DH, col 0,
43144 000040CE B402                         mov     ah,02h          ; now that we can trash the cursor position
43145 000040D0 CD10                         int     10h             ;
43146 000040D2 BA[C352]                     mov     dx,_$StatusLine
43147 000040D5 E87C09                       call    print           ;
43148 000040D8 B403                         mov     ah,3            ; get cursor position
43149 000040DA CD10                         int     10h             ;
43150 000040DC 80EA02                       sub     dl,2            ;
43151 000040DF 8816[7F4C]                   mov     [bLastCol],dl   ; save column where status char will go
43152
43153                                 menu_nostatus:
43154 000040E3 BB0100                       mov     bx,1            ; now prepare to display all the menu items
43155                                 menu_disploop:
43156 000040E6 E8B002                       call    print_item ; print item #BL
43157 000040E9 43                           inc     bx              ; why "inc bx"? because it's a 1-byte opcode
43158 000040EA 3A1E[874C]                   cmp     bl,[bMaxBlock]  ; all done?
43159 000040EE 76F6                         jbe short menu_disploop ; not yet
43160
43161                                 ;   Set cursor position to just below the menu items
43162
43163 000040F0 B200                         mov     dl,0            ; select column
43164 000040F2 88DE                         mov     dh,bl           ;
43165 000040F4 80C604                       add     dh,4            ; select row below menu
43166 000040F7 8A3E[7D4C]                   mov     bh,[bMenuPage]  ;
43167 000040FB B402                         mov     ah,02h          ; set cursor position beneath the block list
43168 000040FD CD10                         int     10h             ;
43169
43170 000040FF BA[B052]                     mov     dx,_$MenuPrmpt
43171 00004102 E84F09                       call    print           ;
43172 00004105 E82903                       call    select_item     ; make a selection, return # in BX
43173 00004108 BA[7050]                     mov     dx,crlfm
43174 0000410B E84609                       call    print           ;
43175 0000410E FF36[814C]                   push    word [bDisableUI]
43176 00004112 800E[814C]01                 or      byte [bDisableUI],1
43177 00004117 E86704                       call    show_status     ; clear the status line now
43178 0000411A 8F06[814C]                   pop     word [bDisableUI]
43179
43180                                 ;   Now begins the "re-organization" process...
43181
43182                                 menu_autoselect:
43183 0000411E 83FBFF                       cmp     bx,-1 ; 0FFFFh    ; clean boot requested?
43184 00004121 7508                         jne short normal_boot ; no
43185 00004123 E8F105                       call    disable_autoexec; basically, add a /D to the command.com line
43186                                 menu_abort:
43187 00004126 29C9                         sub     cx,cx           ; then immediately exit with 0 config.sys image
43188 00004128 E9E400                       jmp menu_exit     ;
43189
43190                                 normal_boot:
43191 0000412B 83FBFE                       cmp     bx,-2 ; 0FFFEh    ; back to top-level menu?
43192 0000412E 7509                         jne short not_topmenu ; no
43193 00004130 8B0E[5603]                   mov     cx,[count]      ; yes, start all over
43194 00004134 29F6                         sub     si,si           ;
43195 00004136 E9FFFD                       jmp     menu_search
43196
43197                                 not_topmenu:
43198 00004139 80BF[8C4C]4F                 cmp     byte [abBlockType+bx],CONFIG_SUBMENU
43199 0000413E 7510                         jne short not_submenu
43200 00004140 01DB                         add     bx,bx           ;
43201 00004142 8BBF[964C]                   mov     di,[aoffBlockName+bx]
43202 00004146 E8E101                       call    srch_block      ; THIS CANNOT FAIL!
43203 00004149 89FE                         mov     si,di           ;
43204 0000414B 89D9                         mov     cx,bx           ; ES:SI and CX are ready for another round
43205 0000414D E929FE                       jmp     menu_found
43206
43207                                 not_submenu:
43208 00004150 01DB                         add     bx,bx           ; get BX -> name of selected block
43209 00004152 8B9F[964C]                   mov     bx,[aoffBlockName+bx]
43210
43211                                 ;   BX should now either be ZERO (meaning no block has been selected) or
43212                                 ;   the offset relative to ES of the block name to be processed (along with
43213                                 ;   all the "common" lines of course)
43214
43215                                 no_selection:
43216 00004156 891E[884C]                   mov     [offDefBlock],bx; save selection
43217 0000415A 8B0E[5603]                   mov     cx,[count]      ; reset ES:SI and CX for reprocessing
43218 0000415E 29F6                         sub     si,si           ;
43219 00004160 1E                           push    ds              ;
43220 00004161 8E1E[6219]                   mov     ds,[config_wrkseg]; this is where we'll store new config.sys image
43221 00004165 29FF                         sub     di,di           ;
43222
43223                                 ;   ES:SI-> config.sys, DS:DI -> new config.sys workspace
43224                                 ;
43225                                 ;   Work our way through the config.sys image again, this time copying
43226                                 ;   all lines that are (A) "common" lines outside any block or (B) lines
43227                                 ;   within the requested block. Lines inside INCLUDEd blocks are transparently
43228                                 ;   copied by copy_block in a recursive fashion; the amount of recursion is
43229                                 ;   limited by the fact INCLUDE statements are REMed by copy_block as they are
43230                                 ;   processed and by the number of unique INCLUDE stmts in config.sys...
43231                                 ;
43232                                 ;   BUGBUG 20-Mar-1992 JeffPar: If we can figure out the lower bound of the
43233                                 ;   stack we're running on, then we should check it inside copy_block
43234
43235                                 copyblock_loop:
43236 00004167 53                           push    bx              ; save selected block name
43237 00004168 E82F01                       call    copy_block      ; process (named or common) block
43238 0000416B 5B                           pop     bx              ;
43239 0000416C 7232                         jc  short move_config ; hit eof
43240
43241                                 ;   copy_block can only return for two reasons: it hit eof or a new block
```

```
43242
43243                                   copyblock_begin:
43244
43245                                   ; 10/09/2023
43246                                   %if 0
43247                                           push    ax              ;
43248                                           push    cx              ;
43249                                           push    si              ;
43250                                           push    di              ; always do "common" blocks
43251                                           mov     di,szCommon
43252                                           push    ds              ;
43253                                           push    cs              ;
43254                                           pop     ds              ;
43255                                           call    comp_names      ;
43256                                           pop     ds              ;
43257                                           pop     di              ;
43258                                           pop     si              ;
43259                                           pop     cx              ;
43260                                           pop     ax              ;
43261                                           je  short copyblock_check
43262                                   %endif
43263                                       ; 10/09/2023
43264 0000416E 57                          push    di
43265 0000416F BF[CB4C]                    mov     di,szCommon    ; always do "common" blocks
43266 00004172 E81602                      call    comp_names_x   ; (comp_names_safe)
43267 00004175 5F                          pop     di
43268 00004176 740F                        je      short copyblock_check
43269
43270 00004178 09DB                            or      bx,bx           ; is there a block name to check?
43271 0000417A 7414                            jz  short copyblock_skip ; no
43272 0000417C 57                              push    di              ;
43273 0000417D 89DF                            mov     di,bx           ; check block against given block name
43274 0000417F 1E                              push    ds              ;
43275 00004180 06                              push    es              ;
43276 00004181 1F                              pop     ds              ;
43277 00004182 E8E501                          call    comp_names      ; is this the block we really want to do?
43278 00004185 1F                              pop     ds              ;
43279 00004186 5F                              pop     di              ;
43280                                   copyblock_check:
43281 00004187 7217                            jc  short move_config ; hit eof
43282 00004189 7505                            jne short copyblock_skip ;
43283 0000418B E85A04                          call    skip_opt_line   ;
43284 0000418E EBD7                            jmp short copyblock_loop
43285
43286                                   copyblock_skip:                 ;
43287 00004190 E85504                          call    skip_opt_line   ; this ain't the block we wanted, so skip it
43288 00004193 E87804                          call    get_char        ;
43289 00004196 7208                            jc  short move_config ; hit eof
43290 00004198 247F                            and     al,~CONFIG_OPTION_QUERY ; 7Fh
43291 0000419A 3C5B                            cmp     al,CONFIG_BEGIN ;
43292 0000419C 74D0                            je  short copyblock_begin
43293 0000419E EBF0                            jmp     short copyblock_skip ; anything else is just skipped
43294                                   ;
43295                                   ;   To create as little risk to the rest of SysInit as little as possible,
43296                                   ;   and to free the workspace at "config_wrkseg" for creating an environment,
43297                                   ;   copy the new config.sys image to "confbot"
43298                                   ;
43299                                   move_config:
43300 000041A0 89F9                            mov     cx,di           ; now copy workspace at DS:DI to "confbot"
43301 000041A2 51                              push    cx              ;
43302                                   ;
43303                                   ;   But first, copy the CONFIG=<configuration><0> string to the workspace,
43304                                   ;   since the configuration name only currently exists in the "confbot" area
43305                                   ;
43306                                       ;mov    cx,7
43307 000041A3 B90700                      mov     cx,szMenu-szBoot-1
43308 000041A6 BE[BE4C]                        mov     si,szBoot ; first copy the CONFIG= part
43309 000041A9 47                              inc     di              ; skip a byte, in case absolutely nothing
43310                                                                  ; was copied to the workspace, because we always
43311                                                                  ; zero the first byte of the workspace (below)
43312                                   copy_boot:
43313                                       ;lods    byte ptr cs:[si];
43314 000041AA 2E                              cs
43315 000041AB AC                          lodsb
43316 000041AC 8805                        mov     [di],al         ;
43317 000041AE 47                              inc     di              ;
43318 000041AF E2F9                            loop    copy_boot       ;
43319
43320 000041B1 06                              push    es              ; then copy the configuration name
43321                                          ;mov       cx,128-7      ; put an upper limit on the name, to be safe
43322                                       ; 04/01/2023
43323 000041B2 B179                        mov     cl,128-7
43324 000041B4 2E8B36[884C]                mov     si,[cs:offDefBlock]; ES:SI -> default block name
43325 000041B9 09F6                            or      si,si           ; valid?
43326 000041BB 7505                            jnz short l1       ; yes
43327 000041BD 0E                              push    cs              ;
43328 000041BE 07                              pop     es              ;
43329 000041BF BE[CB4C]                        mov     si,szCommon
43330 000041C2 268A04                  l1:     mov     al,[es:si]      ;
43331 000041C5 E8B205                          call    any_delim       ;
43332 000041C8 7406                            je  short l2       ;
43333 000041CA 8805                            mov     [di],al         ;
43334 000041CC 46                              inc     si              ;
43335 000041CD 47                              inc     di              ;
43336 000041CE E2F2                            loop    l1              ;
43337 000041D0 C6050A                  l2:     mov     byte [di],lf     ; terminate the configuration string
43338 000041D3 07                              pop     es              ;
43339
43340                                   ;   Now we can copy "config_wrkseg" (DS) to "confbot" (ES)
43341
43342 000041D4 29FF                            sub     di,di           ;
43343 000041D6 2E893E[6019]                    mov     [cs:config_envlen],di
43344 000041DB 29F6                            sub     si,si           ;
43345 000041DD 59                              pop     cx              ; recover the size of "config_wrkseg"
43346
43347 000041DE 51                              push    cx              ;
43348 000041DF F3A4                            rep     movsb           ; moved!
43349 000041E1 59                              pop     cx              ;
43350 000041E2 8CD8                            mov     ax,ds           ;
43351 000041E4 1F                              pop     ds              ;
43352
43353                                   ;   Now that the config_wrkseg is available once again, we shall
43354                                   ;   use it to create an environment. The first thing to go in will be
43355                                   ;   the "CONFIG=configuration" thing. It is also important to zero
43356                                   ;   the first byte of the workspace, so that copy_envvar knows the buffer
43357                                   ;   is empty.
43358
43359 000041E5 06                              push    es              ;
43360 000041E6 8EC0                            mov     es,ax           ;
43361 000041E8 46                              inc     si              ; ES:SI -> "CONFIG=configuration"
43362 000041E9 26C606000000                    mov     byte [es:0],0    ;empty the environment block
43363 000041EF E82600                          call    copy_envvar     ; copy envvar at ES:SI to "config_wrkseg"
43364 000041F2 07                              pop     es
43365
```

```
                                ;   Before returning, restore the default video page setting but do NOT
                                ;   do it using INT 10h's Set Active Page function, because if the menu was
                                ;   displayed on a different page, then it's because we don't want to see
                                ;   all the device driver/TSR goop (which goes to the default page)

                                menu_done:
000041F3 803E[7D4C]00                   cmp     byte [bMenuPage],0
000041F8 7415                           je  short menu_exit       ;
000041FA 06                             push    es              ;
000041FB B84000                         mov     ax,40h          ;
000041FE 8EC0                           mov     es,ax           ;
00004200 A1[834C]                       mov     ax,[wCRTStart]  ;
00004203 26A34E00                       mov     [es:004Eh],ax   ;
00004207 A0[824C]                       mov     al,[bCRTPage]   ;
0000420A 26A26200                       mov     [es:0062h],al   ;
0000420E 07                             pop     es              ;
                                menu_exit:
0000420F 890E[5603]                     mov     [count],cx      ; set new counts
00004213 890E[5803]                     mov     [org_count],cx  ;
                                        ; 10/09/2023 (*) - Erdogan Tan
                                    ; MSDOS 6.21 IO.SYS - SYSINIT:46D3h
                                    ; PCDOS 7.1 IBMBIO.COM - SYSINIT:491Ah
                                    ;sub    si,si           ; always return ES:SI pointing to config.sys
00004217 C3                         retn

                                ; (*) NOTE: MSDOS 6.0 source code (SYSINIT2.ASM) contains 'sub si,si' at this
                                ;   position (then 'retn' just after it)
                                ;   but MSDOS 6.21 and PCDOS 7.1 SYSINITs contain only 'retn' here.

                                ;--------------------------------------------------------------------------
                                ;
                                ;   copy_envvar:  copy the envvar at ES:SI to "config_wrkseg"
                                ;
                                ;   INPUT
                                ;    ES:SI -> environment variable (in the form "var=string<cr/lf>")
                                ;
                                ;   OUTPUT
                                ;       config_envlen (ie, where to put next envvar) updated appropriately
                                ;       carry set if error (eg, missing =); clear otherwise
                                ;
                                ;   OTHER REGS USED
                                ;       None
                                ;
                                ;   NOTES
                                ;       None
                                ;
                                ;   HISTORY
                                ;       Created 29-Mar-1992 by JeffPar
                                ;
                                ;--------------------------------------------------------------------------

                                    ; 04/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
                                    ; (SYSINIT:46D4h)

                                copy_envvar:
00004218 51                             push    cx              ;
00004219 56                             push    si              ;
0000421A 1E                             push    ds              ;
0000421B 06                             push    es              ;
0000421C 06                             push    es              ;
0000421D 8E06[6219]                     mov     es,[config_wrkseg] ; ES:DI to point to next available byte
00004221 1F                             pop     ds                 ; DS:SI to point to envvar

                                ;   Have to calculate the length of the variable name (and if we hit
                                ;   the end of the line before we hit '=', then it's curtains for this
                                ;   config.sys line)
                                ;
                                ;   The check for NULL is important because copy_envvar is also used to copy
                                ;   the initial CONFIG= setting, which will have been zapped by a NULL if no
                                ;   menu block existed (in order to prevent the creation of an environment)

00004222 29C9                           sub     cx,cx           ;
                                copy_varlen:                    ;
00004224 AC                             lodsb                   ;
00004225 08C0                           or      al,al           ; NULL?
                                        ;stc       ; 10/09/2023 (x)
00004227 746B                           jz  short copy_envexit ; yes, abort
00004229 3C0D                           cmp     al,cr           ;
                                        ;stc       ; 10/09/2023 (x)
0000422B 7467                           je  short copy_envexit
0000422D 3C0A                           cmp     al,lf           ;
                                        ;stc       ; 10/09/2023 (x)
0000422F 7463                           je  short copy_envexit
00004231 41                             inc     cx              ;
00004232 3C3D                           cmp     al,'='          ;
00004234 75EE                           jne short copy_varlen
00004236 B000                           mov     al,0            ;
00004238 8A24                           mov     ah,[si]         ; save char after '='
0000423A 29CE                           sub     si,cx           ; back up to given varname
0000423C 49                             dec     cx              ; CX == # of bytes in varname
0000423D 29FF                           sub     di,di           ; start looking for DS:SI at ES:0
                                copy_varsrch:
0000423F 263805                         cmp     byte [es:di],al
00004242 7425                           je  short copy_envprep ; search failed, just copy var
00004244 89FB                           mov     bx,di           ; ES:BX -> start of this varname
00004246 51                             push    cx              ;
00004247 56                             push    si              ;
00004248 F3A6                           repe    cmpsb           ;
0000424A 5E                             pop     si              ;
0000424B 59                             pop     cx              ;
0000424C 7531                           jne short copy_varnext ; no match, skip to next varname
0000424E 26803D3D                       cmp     byte [es:di],'='
00004252 752B                           jne     short copy_varnext ; no match, there's more characters

                                ;   Previous occurrence of variable has been found; determine the
                                ;   entire length and then destroy it

00004254 B9FFFF                         mov     cx,-1           ;
00004257 F2AE                           repne   scasb           ; guaranteed to get null (since we put it there)
00004259 56                             push    si              ;
0000425A 89FE                           mov     si,di           ;
0000425C 89DF                           mov     di,bx           ;
0000425E 2E8B0E[6019]                   mov     cx,[cs:config_envlen]
00004263 29F1                           sub     cx,si           ; destroy variable now
                                    ;rep movs byte ptr es:[di],byte ptr es:[si]
                                    ;;db 0F3h,26h,0A4h ; MSDOS 6.21 IO.SYS - SYSINIT:4724h

00004265 F3                         rep     ; 0F3h
00004266 26                         es      ; 26h
00004267 A4                         movsb   ; 0A4h

00004268 5E                         pop     si
                                copy_envprep:
00004269 80FC0D                         cmp     ah,cr           ; if there is nothing after the '='
```

```
43490 0000426C 741D                     je  short copy_envdel ; then just exit with variable deleted
43491 0000426E 80FC0A                   cmp     ah,lf           ;
43492 00004271 7418                     je  short copy_envdel
43493                                   ;jmp       short copy_envloop
43494                                 ; 04/01/2023
43495                               copy_envloop:                 ;
43496 00004273 AC                       lodsb                   ;
43497 00004274 3C0D                     cmp     al,cr           ;
43498 00004276 7410                     je      short copy_envdone
43499 00004278 3C0A                     cmp     al,lf           ;
43500 0000427A 740C                     je      short copy_envdone
43501 0000427C AA                       stosb                   ;
43502 0000427D EBF4                     jmp     short copy_envloop
43503
43504                               copy_varnext:                 ;
43505 0000427F 51                       push    cx              ;
43506 00004280 B9FFFF                   mov     cx,-1           ;
43507 00004283 F2AE                     repne   scasb           ;
43508 00004285 59                       pop     cx              ;
43509 00004286 EBB7                     jmp short copy_varsrch
43510
43511                                   ; 04/01/2023
43512                               ;copy_envloop:                ;
43513                               ;   lodsb                   ;
43514                               ;   cmp     al,cr           ;
43515                               ;   je      short copy_envdone
43516                               ;   cmp     al,lf           ;
43517                               ;   je      short copy_envdone
43518                               ;   stosb                   ;
43519                               ;   jmp     short copy_envloop
43520
43521                               copy_envdone:                 ;
43522 00004288 28C0                     sub     al,al           ; do SUB to clear carry as well
43523 0000428A AA                       stosb                   ; always null-terminate these puppies
43524                               copy_envdel:                  ;
43525 0000428B 268805                   mov     [es:di],al      ; and stick another null to terminate the env.
43526 0000428E 2E893E[6019]             mov     [cs:config_envlen],di
43527                                 ; 10/09/2023 (x) - Erdogan Tan
43528 00004293 F9                       stc ; in order to clear carry flag via cmc (compact code trick!)
43529                               copy_envexit:                 ;
43530 00004294 F5                       cmc ; (x) ; reverse carry flag status (je -> cf=1)
43531 00004295 07                       pop     es              ;
43532 00004296 1F                       pop     ds              ;
43533 00004297 5E                       pop     si              ;
43534 00004298 59                       pop     cx              ;
43535
43536                               copy_done:  ; 18/12/2022
43537 00004299 C3                       retn
43538
43539                               ;--------------------------------------------------------------------------
43540                               ;
43541                               ;   copy_block:  copy the current block to the new config.sys workspace
43542                               ;
43543                               ;   INPUT
43544                               ;       CX == remaining bytes in "organized" config.sys memory image
43545                               ;    ES:SI -> remaining bytes in "organized" config.sys memory image
43546                               ;    DS:DI -> new config.sys workspace (equal in size to the original
43547                               ;             config.sys image) where the current block is to be copied
43548                               ;
43549                               ;   OUTPUT
43550                               ;       Same as above
43551                               ;       AL also equals the last character read from the organized image
43552                               ;
43553                               ;   OTHER REGS USED
43554                               ;       All
43555                               ;
43556                               ;   NOTES
43557                               ;       None
43558                               ;
43559                               ;   HISTORY
43560                               ;       Created 16-Mar-1992 by JeffPar
43561                               ;
43562                               ;--------------------------------------------------------------------------
43563
43564                                   ; 04/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
43565                                   ; (SYSINIT:4759h)
43566
43567                               copy_block:
43568 0000429A E87103                   call    get_char        ; check for include
43569 0000429D 72FA                     jc  short copy_done     ;
43570 0000429F 247F                   and     al,~CONFIG_OPTION_QUERY ; 7Fh
43571 000042A1 3C5B                   cmp     al,CONFIG_BEGIN ; another BEGIN implies END as well
43572 000042A3 74F4                     je  short copy_done ;
43573
43574 000042A5 3C4A                     cmp     al,CONFIG_INCLUDE ; 'J'
43575 000042A7 88E0                     mov     al,ah           ; AL == the original line code
43576 000042A9 753A                     jne short copy_line     ; not an "include" line
43577
43578                               ;   We have hit an "INCLUDE" line; first, REM out the line so that we
43579                               ;   never try to include the block again (no infinite include loops please),
43580                               ;   then search for the named block and call copy_block again.
43581
43582 000042AB 26C644FF30                   mov     byte [es:si-1],CONFIG_REM ; '0'
43583 000042B0 57                       push    di              ;
43584
43585 000042B1 BF[C64C]                 mov     di,szMenu
43586 000042B4 E8D400                   call    comp_names_safe ; don't allow INCLUDE MENU
43587 000042B7 7426                     je  short copy_skip     ;
43588
43589 000042B9 BF[CB4C]                 mov     di,szCommon
43590 000042BC E8CC00                   call    comp_names_safe ; don't allow INCLUDE COMMON
43591 000042BF 741E                     je  short copy_skip     ;
43592
43593 000042C1 89F7                     mov     di,si           ; try to find the block
43594 000042C3 E86400                   call    srch_block      ;
43595 000042C6 89FA                     mov     dx,di           ;
43596                                 ; 10/09/2023
43597                                 ;pop    di              ;
43598 000042C8 7514                     jne short copy_error ; no such block
43599 000042CA 5F                     pop     di  ; 10/09/2023
43600 000042CB 51                       push    cx              ;
43601 000042CC 89D9                     mov     cx,bx           ;
43602 000042CE 56                       push    si              ;
43603 000042CF 4A                       dec     dx              ;
43604 000042D0 89D6                     mov     si,dx           ;
43605 000042D2 E80E03                   call    skip_line       ; skip the rest of the "block name" line
43606 000042D5 E8C2FF                   call    copy_block      ; and copy in the rest of that block
43607 000042D8 5E                       pop     si              ;
43608 000042D9 59                       pop     cx              ;
43609 000042DA 28C0                     sub     al,al           ; force skip_opt_line to skip...
43610 000042DC EB2B                     jmp     short copy_nextline
43611
43612                               copy_error:
43613                                   ; 10/09/2023
```

```
43614 000042DE F8                       clc
43615                                 copy_skip:
43616 000042DF 5F                       pop     di
43617                                 ;copy_error:
43618                                   ; 10/09/2023 (cf=0)
43619                                   ;clc                    ;
43620 000042E0 E80004                   call    print_error     ; note that carry is clear, no pause
43621 000042E3 EB24                     jmp     short copy_nextline
43622
43623                                 ;   Copy the line at ES:SI to the current location at DS:DI
43624
43625                                 copy_line:
43626 000042E5 8805                     mov     [di],al         ;
43627 000042E7 47                       inc     di              ;
43628 000042E8 3C20                     cmp     al,' '          ; is this is a "real" line with a "real" code?
43629 000042EA 721D                     jb  short copy_nextline ; no
43630 000042EC 2E803E[6519]00           cmp     byte [cs:config_multi],0
43631 000042F2 7409                     je  short copy_loop       ; not a multi-config config.sys, don't embed #s
43632 000042F4 E81700                   call    get_linenum       ; BX == line # of line @ES:SI
43633 000042F7 891D                     mov     [di],bx         ; stash it immediately following the line code
43634 000042F9 47                       inc     di              ;
43635 000042FA 47                       inc     di              ;
43636 000042FB EB08                     jmp     short copy_next ;
43637                                 copy_loop:                ;
43638 000042FD E80E03                   call    get_char        ;
43639 00004300 7297                     jc  short copy_done ; end of file
43640 00004302 8805                     mov     [di],al         ;
43641 00004304 47                       inc     di              ;
43642                                 copy_next:
43643 00004305 3C0A                     cmp     al,lf ; 0Ah       ; done with line?
43644 00004307 75F4                     jne short copy_loop       ; nope
43645
43646                                 copy_nextline:
43647 00004309 E8DC02                   call    skip_opt_line   ;
43648 0000430C EB8C                     jmp     short copy_block
43649
43650                                   ; 18/12/2022
43651                                 ;copy_done:
43652                                   ;retn
43653
43654                                 ;--------------------------------------------------------------------------
43655                                 ;
43656                                 ;   get_linenum:  return line # (in BX) of current line (@ES:SI)
43657                                 ;
43658                                 ;   INPUT
43659                                 ;    ES:SI -> some line in the config.sys memory image
43660                                 ;
43661                                 ;   OUTPUT
43662                                 ;       BX == line # (relative to 1)
43663                                 ;
43664                                 ;   OTHER REGS USED
43665                                 ;       DX
43666                                 ;
43667                                 ;   NOTES
43668                                 ;       None
43669                                 ;
43670                                 ;   HISTORY
43671                                 ;       Created 16-Mar-1992 by JeffPar
43672                                 ;
43673                                 ;--------------------------------------------------------------------------
43674
43675                                 get_linenum:
43676 0000430E 50                       push    ax              ;
43677 0000430F 29DB                     sub     bx,bx           ; BX == line # (to be returned)
43678 00004311 51                       push    cx              ;
43679 00004312 89F2                     mov     dx,si           ; DX == the offset we're looking for
43680 00004314 56                       push    si              ;
43681 00004315 2E8B0E[5603]             mov     cx,[cs:count]   ;
43682 0000431A 29F6                     sub     si,si           ; prepare to scan entire file
43683                                 get_linenum_loop:         ;
43684 0000431C E8C402                   call    skip_line       ;
43685 0000431F 7205                     jc  short get_linenum_done
43686 00004321 43                       inc     bx              ;
43687 00004322 39D6                     cmp     si,dx           ; have we exceeded the desired offset yet?
43688 00004324 72F6                     jb  short get_linenum_loop ; no
43689                                 get_linenum_done:         ;
43690 00004326 5E                       pop     si              ;
43691 00004327 59                       pop     cx              ;
43692 00004328 58                       pop     ax              ;
43693 00004329 C3                       retn
43694
43695                                 ;--------------------------------------------------------------------------
43696                                 ;
43697                                 ;   srch_block:  searches entire config.sys for block name @ES:DI
43698                                 ;
43699                                 ;   INPUT
43700                                 ;       ES -> config.sys image
43701                                 ;    ES:DI -> block name to find
43702                                 ;
43703                                 ;   OUTPUT
43704                                 ;       ZF flag set, if found
43705                                 ;    ES:DI -> just past the name in the block heading, if found
43706                                 ;       BX == # bytes remaining from that point, if found
43707                                 ;
43708                                 ;   OTHER REGS USED
43709                                 ;       None
43710                                 ;
43711                                 ;   NOTES
43712                                 ;       This differs from "find_block" in that it searches the ENTIRE
43713                                 ;       config.sys image, not merely the remaining portion, and that it
43714                                 ;       takes a pointer to block name that is *elsewhere* in the image
43715                                 ;       (ie, ES) as opposed to some string constant in our own segment (DS).
43716                                 ;
43717                                 ;   HISTORY
43718                                 ;       Created 16-Mar-1992 by JeffPar
43719                                 ;
43720                                 ;--------------------------------------------------------------------------
43721
43722                                 srch_block:           ; returns BX -> named block in CONFIG.SYS
43723 0000432A 50                       push    ax              ;
43724 0000432B 51                       push    cx              ;
43725 0000432C 2E8B0E[5603]             mov     cx,[cs:count]   ;
43726 00004331 56                       push    si              ;
43727 00004332 29F6                     sub     si,si           ;
43728 00004334 1E                       push    ds              ;
43729 00004335 06                       push    es              ;
43730 00004336 1F                       pop     ds              ;
43731 00004337 E80900                   call    find_block      ;
43732 0000433A 89F7                     mov     di,si           ;
43733 0000433C 89CB                     mov     bx,cx           ;
43734 0000433E 1F                       pop     ds              ;
43735 0000433F 5E                       pop     si              ;
43736 00004340 59                       pop     cx              ;
43737 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
```

```
43862                                   ;
43863                                   ;   OUTPUT
43864                                   ;       Menu item displayed, with appropriate highlighting if BL == bDefBlock
43865                                   ;
43866                                   ;   OTHER REGS USED
43867                                   ;       None
43868                                   ;
43869                                   ;   NOTES
43870                                   ;       This function saves/restores the current cursor position, so you
43871                                   ;       needn't worry about it.
43872                                   ;
43873                                   ;   HISTORY
43874                                   ;       Created 16-Mar-1992 by JeffPar
43875                                   ;
43876                                   ;--------------------------------------------------------------------------
43877
43878                                       ; 04/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
43879                                       ; (SYSINIT:485Ah)
43880
43881                                   print_item:       ; prints menu item #BL (1 to N)
43882 00004399 50                             push    ax              ;
43883 0000439A 53                             push    bx              ;
43884 0000439B 51                             push    cx              ;
43885 0000439C 52                             push    dx              ;
43886 0000439D 56                             push    si              ;
43887 0000439E B403                           mov     ah,03h          ; get cursor position
43888 000043A0 8A3E[7D4C]                     mov     bh,[bMenuPage]  ; always page zero
43889 000043A4 CD10                           int     10h             ; DH/DL = row/column
43890 000043A6 52                             push    dx              ; save it
43891 000043A7 B402                           mov     ah,02h          ; set cursor position
43892 000043A9 88DE                           mov     dh,bl           ;
43893 000043AB 80C603                         add     dh,3            ;
43894 000043AE B205                           mov     dl,5            ;
43895 000043B0 CD10                           int     10h             ; set cursor position for correct row/col
43896 000043B2 88D8                           mov     al,bl           ;
43897 000043B4 0430                           add     al,'0'          ; convert menu item # to ASCII digit
43898 000043B6 8A26[7C4C]                     mov     ah,[bMenuColor] ; normal attribute
43899 000043BA 3A1E[864C]                     cmp     bl,[bDefBlock]  ; are we printing the current block?
43900 000043BE 7510                           jne short print_other ; no
43901 000043C0 80CC70                         or      ah,70h          ; yes, set bgnd color to white
43902 000043C3 88E5                           mov     ch,ah           ;
43903 000043C5 B104                           mov     cl,4            ;
43904 000043C7 D2C5                           rol     ch,cl           ;
43905 000043C9 38E5                           cmp     ch,ah           ; are fgnd/bgnd the same?
43906 000043CB 7503                           jne short print_other ; no
43907 000043CD 80F408                         xor     ah,08h          ; yes, so modify the fgnd intensity
43908                                   print_other:                    ;
43909 000043D0 B700                           mov     bh,0            ;
43910 000043D2 01DB                           add     bx,bx           ;
43911 000043D4 8BBF[AA4C]                     mov     di,[aoffBlockDesc+bx]
43912 000043D8 88E3                           mov     bl,ah           ; put the attribute in the correct register now
43913 000043DA 8A3E[7D4C]                     mov     bh,[bMenuPage]  ; get correct video page #
43914 000043DE B409                           mov     ah,09h          ; write char/attr
43915 000043E0 B90100                         mov     cx,1            ;
43916 000043E3 CD10                           int     10h             ;
43917 000043E5 FEC2                           inc     dl              ; increment column
43918 000043E7 B402                           mov     ah,02h          ;
43919 000043E9 CD10                           int     10h             ;
43920                                         ;mov       ax,0900h+'.'    ;
43921 000043EB B82E09                         mov ax,092Eh
43922 000043EE CD10                       int     10h             ; display '.'
43923 000043F0 FEC2                           inc     dl              ; increment column
43924 000043F2 B402                           mov     ah,02h          ;
43925 000043F4 CD10                           int     10h             ;
43926                                         ;mov       ax,0900h+' '    ;
43927 000043F6 B82009                         mov ax,0920h
43928 000043F9 CD10                       int     10h             ; display ' '
43929 000043FB FEC2                           inc     dl              ; increment column
43930 000043FD B402                           mov     ah,02h          ;
43931 000043FF CD10                           int     10h             ;
43932 00004401 06                             push    es              ;
43933                                   print_loop:                     ;
43934 00004402 268A05                         mov     al,[es:di]; get a character of the description
43935 00004405 47                             inc     di              ;
43936 00004406 3C09                           cmp     al,TAB ; 9; substitute spaces for tabs
43937 00004408 7502                           jne short print_nontab ;
43938 0000440A B020                           mov     al,' '          ;
43939                                   print_nontab:                   ;
43940 0000440C 3C20                           cmp     al,' '          ;
43941 0000440E 7215                           jb  short print_done ; stop at the 1st character < space
43942 00004410 3C24                           cmp     al,'$'          ;
43943 00004412 7411                           je  short print_done ; also stop on $
43944 00004414 B409                           mov     ah,09h          ; display function #
43945 00004416 CD10                           int     10h             ;
43946 00004418 FEC2                           inc     dl              ; increment column
43947 0000441A 80FA4E                         cmp     dl,78           ; far enough?
43948 0000441D 7306                           jae short print_done ; yes
43949 0000441F B402                           mov     ah,02h          ;
43950 00004421 CD10                           int     10h             ;
43951 00004423 EBDD                           jmp short  print_loop
43952                                   print_done:                     ;
43953 00004425 07                             pop     es              ;
43954 00004426 5A                             pop     dx              ;
43955 00004427 B402                           mov     ah,02h          ;
43956 00004429 CD10                           int     10h             ; restore previous row/col
43957 0000442B 5E                             pop     si              ;
43958 0000442C 5A                             pop     dx              ;
43959 0000442D 59                             pop     cx              ;
43960 0000442E 5B                             pop     bx              ;
43961 0000442F 58                             pop     ax              ;
43962 00004430 C3                             retn                      ;
43963
43964                                   ;--------------------------------------------------------------------------
43965                                   ;
43966                                   ;   select_item:  wait for user to select menu item, with time-out
43967                                   ;
43968                                   ;   INPUT
43969                                   ;       None
43970                                   ;
43971                                   ;   OUTPUT
43972                                   ;       BX == menu item # (1-N), or -1 for clean boot
43973                                   ;       Selected menu item highlighted
43974                                   ;       Cursor positioned beneath menu, ready for tty-style output now
43975                                   ;
43976                                   ;   OTHER REGS USED
43977                                   ;       None
43978                                   ;
43979                                   ;   NOTES
43980                                   ;       None
43981                                   ;
43982                                   ;   HISTORY
43983                                   ;       Created 16-Mar-1992 by JeffPar
43984                                   ;
43985                                   ;--------------------------------------------------------------------------
```

```
43986
43987                                   select_item:            ; returns digit value in BX (trashes AX/CX/DX)
43988 00004431 8A1E[864C]                       mov     bl,[bDefBlock]  ; BL will be the default block #
43989 00004435 88D8                             mov     al,bl           ;
43990 00004437 E83701                           call    disp_num        ;
43991 0000443A E84401                           call    show_status     ; display current interactive status
43992 0000443D 803E[8A4C]FF                     cmp     byte [secTimeOut],-1
43993 00004442 7452                             je  short input_key       ; no time-out, just go to input
43994 00004444 B42C                             mov     ah,GET_TIME ; 2Ch
43995 00004446 CD21                             int     21h             ;
43996 00004448 88F7                             mov     bh,dh           ; BH = initial # of seconds
43997                                   check_time:
43998 0000444A A0[8A4C]                         mov     al,[secTimeOut] ;
43999 0000444D 2A06[8B4C]                       sub     al,[secElapsed] ;
44000 00004451 730D                             jae short show_time       ;
44001 00004453 800E[854C]02                     or      byte [bQueryOpt],2  ; disable all further prompting
44002 00004458 C606[8B4C]00                     mov     byte [secElapsed],0
44003 0000445D E9F600                           jmp select_done    ; time's up!
44004                                   show_time:
44005 00004460 53                               push    bx              ;
44006 00004461 88C3                             mov     bl,al           ; save # in BL
44007 00004463 8A3E[7D4C]                       mov     bh,[bMenuPage]  ;
44008 00004467 B403                             mov     ah,03h          ; get cursor position
44009 00004469 CD10                             int     10h             ;
44010 0000446B 52                               push    dx              ;
44011 0000446C 80C208                       add     dl,8            ; move cursor to the right
44012 0000446F B402                             mov     ah,02h          ; set cursor position
44013 00004471 CD10                             int     10h             ;
44014 00004473 BA[2753]                         mov     dx,_$TimeOut
44015 00004476 E8DB05                           call    print           ; print the "Time remaining: " prompt
44016 00004479 88D8                             mov     al,bl           ; recover # from BL
44017 0000447B 98                               cbw                     ; this works because AL is always <= 90
44018 0000447C B10A                             mov     cl,10           ;
44019 0000447E F6F1                             div     cl              ; AL = tens digit, AH = ones digit
44020 00004480 88E1                             mov     cl,ah           ;
44021 00004482 0430                             add     al,'0'          ;
44022 00004484 B40E                             mov     ah,0Eh          ;
44023 00004486 CD10                             int     10h             ; write TTY tens digit
44024 00004488 88C8                             mov     al,cl           ;
44025 0000448A 0430                             add     al,'0'          ;
44026 0000448C B40E                             mov     ah,0Eh          ;
44027 0000448E CD10                             int     10h             ; write TTY ones digit
44028 00004490 5A                               pop     dx
44029 00004491 B402                             mov     ah,02h          ; set cursor position back to where it was
44030 00004493 CD10                             int     10h             ;
44031 00004495 5B                               pop     bx              ;
44032                                   input_key:
44033 00004496 B406                             mov     ah,RAW_CON_IO ; 6
44034 00004498 B2FF                             mov     dl,0FFh         ; input request
44035 0000449A CD21                             int     21h             ;
44036 0000449C 751F                             jnz short got_key  ;
44037 0000449E 803E[8A4C]FF                     cmp     byte [secTimeOut],-1; is there a time-out?
44038 000044A3 74F1                             je  short input_key       ; no, just go back to input
44039 000044A5 B42C                             mov     ah,GET_TIME     ;
44040 000044A7 CD21                             int     21h             ; DH = seconds
44041 000044A9 88F4                             mov     ah,dh           ;
44042 000044AB 28FE                             sub     dh,bh           ; should generally be zero or one
44043 000044AD 88E7                             mov     bh,ah           ;
44044 000044AF 7302                             jnc short got_time ;
44045 000044B1 B601                             mov     dh,1            ; it wrapped back to zero, so assume one
44046                                   got_time:
44047 000044B3 08F6                             or      dh,dh           ; any change?
44048 000044B5 74DF                             jz  short input_key       ; no
44049 000044B7 0036[8B4C]                       add     [secElapsed],dh ;
44050 000044BB EB8D                             jmp short check_time ;
44051                                   got_key:
44052 000044BD 50                               push    ax              ;
44053 000044BE B8FFFF                           mov     ax,-1           ; zap both secTimeOut and secElapsed
44054 000044C1 8706[8A4C]                       xchg    [secTimeOut],ax
44055 000044C5 3CFF                             cmp     al,-1           ; was time-out already disabled?
44056 000044C7 740E                             je  short timeout_disabled ; yes
44057 000044C9 53                               push    bx              ; let's disable # seconds display
44058 000044CA B8200A                           mov     ax,0A20h        ; write multiple spaces
44059 000044CD 8B1E[7C4C]                       mov     bx,[bMenuColor]
44060 000044D1 B95000                           mov     cx,80           ; 80 of them, to be safe
44061 000044D4 CD10                             int     10h             ; to completely obliterate # seconds display
44062 000044D6 5B                               pop     bx                ;
44063
44064                                   timeout_disabled:
44065 000044D7 58                               pop     ax              ;
44066 000044D8 08C0                             or      al,al           ; extended key pressed?
44067 000044DA 755A                             jnz short normal_key ; no
44068 000044DC CD21                             int     21h             ; get the next part of the key then
44069 000044DE 74B6                             jz  short input_key       ; hmmm, what happened to the second part?
44070
44071 000044E0 3C48                             cmp     al,48h          ; up arrow?
44072 000044E2 7510                             jne short not_up   ; no
44073 000044E4 80FB01                           cmp     bl,1            ; are we as up as up can get?
44074 000044E7 76AD                             jbe short input_key       ; yes, ignore it
44075 000044E9 FE0E[864C]                       dec     byte [bDefBlock] ;
44076 000044ED E8A9FE                           call    print_item      ; re-print the current item
44077 000044F0 FECB                             dec     bl              ; and then print the new current item
44078 000044F2 EB12                             jmp     short print1
44079                                   not_up:
44080 000044F4 3C50                             cmp     al,50h          ; down arrow?
44081 000044F6 7518                             jne short not_down ; no
44082 000044F8 3A1E[874C]                       cmp     bl,[bMaxBlock]  ; are we as down as down can get?
44083 000044FC 7310                             jae short to_input_key ; yes, ignore it
44084 000044FE FE06[864C]                       inc     byte [bDefBlock] ;
44085 00004502 E894FE                           call    print_item      ; re-print the current item
44086 00004505 43                               inc     bx              ; and then print the new current item
44087                                   print1:
44088 00004506 88D8                         mov     al,bl           ;
44089                                   print2:
44090 00004508 E88EFE                       call    print_item      ;
44091 0000450B E86300                           call    disp_num        ;
44092                                   to_input_key:
44093 0000450E EB86                             jmp short input_key ; 10/09/2023
44094                                   not_down:
44095 00004510 F606[814C]01                     test    byte [bDisableUI],1
44096 00004515 75F7                             jnz short to_input_key ; don't allow F8 or F5
44097 00004517 3C42                             cmp     al,42h          ; F8 function key?
44098 00004519 750B                             jne short not_f8   ; no
44099 0000451B 8036[854C]01                     xor     byte [bQueryOpt],1
44100 00004520 E85E00                           call    show_status     ;
44101 00004523 E970FF                           jmp     input_key ;
44102                                   not_f8:
44103 00004526 3C3F                             cmp     al,3Fh          ; F5 function key?
44104 00004528 75E4                             jne short to_input_key ; no
44105                                       ; 02/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
44106                                       ; MSDOS 6.21 IO.SYS - SYSINIT:49EBh
44107                                       ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:4C32h)
44108 0000452A 800E[854C]04                 or      byte [bQueryOpt],4 ; no more queries
44109 0000452F BBFFFF                           mov     bx,-1           ; special return code (-1) indicating clean boot
```

```
44110 00004532 B020                     mov     al,' '          ; don't want to display anything really;
44111 00004534 EB26                     jmp     short disp_input ; just want to display the cr/lf sequence...
44112
44113                                 normal_key:
44114 00004536 3C0D                     cmp     al,0Dh          ; Enter?
44115 00004538 741C                     je  short select_done ; yes
44116 0000453A 3C08                     cmp     al,08h          ; backspace?
44117 0000453C 7504                     jne short not_backspace ; no
44118 0000453E BBFEFF                   mov     bx,-2 ; 0FFFEh    ; yes, special return code
44119 00004541 C3                       retn                      ;
44120                                 not_backspace:
44121 00004542 2C30                     sub     al,'0'          ; is greater than '0'?
44122 00004544 76C8                     jbe short to_input_key ; no
44123 00004546 3A06[874C]               cmp     al,[bMaxBlock]  ; is less than or equal to the maximum digit?
44124 0000454A 77C2                     ja  short to_input_key ; no
44125 0000454C A2[864C]                 mov [bDefBlock],al  ;
44126 0000454F E847FE                   call    print_item      ; redisplay the current selection
44127 00004552 88C3                     mov     bl,al           ; set new selection
44128 00004554 EBB2                     jmp short print2
44129
44130                                 select_done:
44131 00004556 B700                     mov     bh,0            ; return a full 16-bit value (for indexing)
44132 00004558 88D8                     mov     al,bl           ;
44133 0000455A 0430                     add     al,'0'          ; convert it into a digit, then display it
44134
44135                                     ; fall into disp_input
44136
44137                                 ; 16/04/2019 - Retro DOS v4.0
44138
44139                                 ;--------------------------------------------------------------------------
44140                                 ;
44141                                 ;   disp_input:  display a single character + cr/lf
44142                                 ;
44143                                 ;   INPUT
44144                                 ;       AL == character to display
44145                                 ;
44146                                 ;   OUTPUT
44147                                 ;       None
44148                                 ;
44149                                 ;   OTHER REGS USED
44150                                 ;       None
44151                                 ;
44152                                 ;   NOTES
44153                                 ;       This function is used not only for the menu input selection but
44154                                 ;       also for the interactive line prompting (the y/n/a thing).
44155                                 ;
44156                                 ;   HISTORY
44157                                 ;       Created 16-Mar-1992 by JeffPar
44158                                 ;
44159                                 ;--------------------------------------------------------------------------
44160
44161
44162                                 disp_input:
44163 0000455C 50                       push    ax
44164                                     ;cmp    al,' '
44165                                     ;jae    short disp_ok
44166                                     ;mov    al,' '
44167                                     ; 10/09/2023 - Retro DOS v4.2 IO:SYS (Optimization)
44168 0000455D B220                     mov     dl,' ' ; 20h
44169 0000455F 38D0                     cmp     al,dl
44170 00004561 7602                     jna     short disp_input_ok
44171                                 disp_ok:
44172 00004563 88C2                     mov     dl,al
44173                                 disp_input_ok:
44174 00004565 B402                     mov     ah,STD_CON_OUTPUT ; 2
44175 00004567 CD21                     int     21h
44176 00004569 BA[7050]                 mov     dx,crlfm
44177 0000456C E8E504                   call    print
44178 0000456F 58                       pop     ax
44179 00004570 C3                       retn
44180
44181                                 ;--------------------------------------------------------------------------
44182
44183                                 disp_num:
44184 00004571 53                       push    bx
44185 00004572 0430                     add     al,'0'
44186 00004574 B40A                     mov     ah,0Ah
44187 00004576 8B1E[7C4C]               mov     bx,[bMenuColor]
44188 0000457A B90100                   mov     cx,1
44189 0000457D CD10                     int     10h
44190 0000457F 5B                       pop     bx
44191 00004580 C3                       retn
44192
44193                                 ;--------------------------------------------------------------------------
44194                                 ;
44195                                 ;   show_status:  display current interactive mode setting (on/off/none)
44196                                 ;
44197                                 ;   INPUT
44198                                 ;       None
44199                                 ;
44200                                 ;   OUTPUT
44201                                 ;       None
44202                                 ;
44203                                 ;   OTHER REGS USED
44204                                 ;       None
44205                                 ;
44206                                 ;   NOTES
44207                                 ;       None
44208                                 ;
44209                                 ;   HISTORY
44210                                 ;       Created 16-Mar-1992 by JeffPar
44211                                 ;
44212                                 ;--------------------------------------------------------------------------
44213
44214                                 show_status:
44215 00004581 53                       push    bx              ; BL = video page #
44216 00004582 8B1E[7C4C]               mov     bx,[bMenuColor]
44217 00004586 B403                     mov     ah,03h          ; get cursor position
44218 00004588 CD10                     int     10h             ;
44219 0000458A 52                       push    dx              ; save it
44220 0000458B B402                     mov     ah,02h          ; set cursor position
44221 0000458D 8B16[7F4C]               mov     dx,[bLastCol]   ; set correct row/col
44222 00004591 F606[814C]01             test    byte [bDisableUI],1
44223 00004596 740C                     jz  short show_onoff ; just show on/off
44224 00004598 B200                     mov     dl,0            ;
44225 0000459A CD10                     int     10h             ;
44226 0000459C B8200A                   mov     ax,0A20h        ; write multiple spaces
44227 0000459F B95000                   mov     cx,80           ; 80 of them, to be exact
44228                                   ; 10/09/2023
44229                                   ;int   10h             ; to obliterate the status line
44230 000045A2 EB11                     jmp     short show_done ;
44231                                 show_onoff:
44232 000045A4 CD10                     int     10h
44233                                       ; - 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
```

```
44482 00004614 AC                       lodsb
44483 00004615 88C4                         mov     ah,al           ;
44484 00004617 C3                           retn                      ;
44485                                 get_fail:                       ; restore CX to zero
44486 00004618 B90000                       mov     cx,0            ; leave carry set, zero not set
44487                                 nearby_ret:
44488 0000461B C3                           retn
44489
44490                                 ;--------------------------------------------------------------------------
44491                                 ;
44492                                 ;   query_user: ask user whether to execute current config.sys command
44493                                 ;
44494                                 ;   INPUT
44495                                 ;       AL == current command code
44496                                 ;    ES:SI -> current command line in config.sys
44497                                 ;    config_cmd == current command code, but with QUERY bit intact
44498                                 ;                  (00h used to generate "Process AUTOEXEC.BAT" prompt)
44499                                 ;
44500                                 ;   OUTPUT
44501                                 ;       CF set if command should be ignored (it is also REM'ed out)
44502                                 ;
44503                                 ;   OTHER REGS USED
44504                                 ;       BX, CX, DX, DI
44505                                 ;
44506                                 ;   NOTES
44507                                 ;       None
44508                                 ;
44509                                 ;   HISTORY
44510                                 ;       Created 16-Mar-1992 by JeffPar
44511                                 ;
44512                                 ;--------------------------------------------------------------------------
44513
44514                                     ; 31/12/2022 - Retro UNIX 386 v4.2 (Modified MSDOS 6.21 IO.SYS)
44515                                     ; (SYSINIT:4AE5h)
44516
44517                                     ; 12/12/2022
44518                                 query_user:
44519 0000461C F606[854C]04                 test    byte [bQueryOpt],4      ; answer no to everything?
44520                                     ; 01/01/2023
44521 00004621 7403                     jz      short qu_1              ;
44522 00004623 E9B900                   jmp     skip_all
44523                                     ; 12/12/2022
44524                                     ;;jmp   short skip_all          ;
44525                                     ;jnz    short skip_all
44526                                 qu_1:
44527 00004626 F606[854C]02             test    byte [bQueryOpt],2      ; answer yes to everything?
44528 0000462B 75EE                         jnz short nearby_ret        ; yes (and return carry clear!)
44529 0000462D 50                           push    ax                  ;
44530 0000462E A0[6419]                     mov     al,[config_cmd]     ;
44531 00004631 F606[854C]01                 test    byte [bQueryOpt],1      ; query every command?
44532 00004636 7506                         jnz short query_all             ; yes
44533 00004638 A880                         test    al,CONFIG_OPTION_QUERY  ;
44534                                     ; 01/01/2023
44535 0000463A 7502                     jnz     short query_all             ;
44536                                     ; 12/12/2022
44537                                     ;;jmp   short do_cmd            ;
44538                                     ;jz     short do_cmd ; cf=0
44539
44540                                     ; 01/01/2023
44541 0000463C 58                       pop     ax
44542 0000463D C3                       retn
44543
44544                                 query_all:
44545
44546                                 ;   Search for the command code (AL) in "comtab", and then print
44547                                 ;   out the corresponding keyword, followed by the rest of the actual
44548                                 ;   line pointed to by ES:SI
44549
44550 0000463E 56                           push    si                      ; save pointer to rest of CONFIG.SYS line
44551 0000463F BA[8253]                     mov     dx,_$AutoPrmpt          ;
44552 00004642 247F                         and     al,~CONFIG_OPTION_QUERY ; 7Fh
44553 00004644 7450                         jz  short generic_prompt  ; config_cmd must have been 0
44554
44555 00004646 88C6                         mov     dh,al                   ; save config_cmd in DH
44556 00004648 29DB                         sub     bx,bx                   ;
44557 0000464A BF[D24C]                     mov     di,comtab           ;
44558                                 find_match:                             ;
44559 0000464D 8A1D                         mov     bl,[di]                 ; get size of current keyword
44560 0000464F 08DB                         or      bl,bl                   ;
44561 00004651 7425                         jz  short line_print        ; end of table
44562 00004653 47                           inc     di                      ;
44563 00004654 3A01                         cmp     al,[di+bx]              ; match?
44564 00004656 7405                         je  short cmd_match             ; yes
44565 00004658 8D7901                       lea     di,[di+bx+1]            ; otherwise, skip this command code
44566                                     ; 13/05/2019
44567 0000465B EBF0                         jmp short find_match        ; loop
44568                                 cmd_match:                              ;
44569 0000465D 8A4DFF                       mov     cl,[di-1]               ;
44570 00004660 B500                         mov ch,0                    ;
44571 00004662 B402                         mov     ah,STD_CON_OUTPUT ; 2
44572                                 cmd_print:                              ;
44573 00004664 8A05                         mov     al,[di]                 ;
44574 00004666 47                           inc     di                      ;
44575 00004667 88C2                         mov     dl,al                   ;
44576 00004669 CD21                         int     21h                     ;
44577 0000466B E2F7                         loop    cmd_print               ;
44578 0000466D B23D                         mov     dl,'='                  ;
44579 0000466F 80FE56                       cmp     dh,CONFIG_SET  ; 'V'    ; for SET commands, don't display a '='
44580 00004672 7502                         jne short cmd_notset        ;
44581 00004674 B220                         mov     dl,' '                  ;
44582                                 cmd_notset:
44583 00004676 CD21                         int     21h                     ; '=' looks funny on SET commands
44584                                 line_print:                             ;
44585                                     ;lods   byte ptr es:[si]        ;
44586 00004678 26                           es
44587 00004679 AC                       lodsb
44588 0000467A 08C0                     or      al,al                   ;
44589 0000467C 7502                         jnz short non_null          ;
44590 0000467E B020                         mov     al,' '                  ;
44591                                 non_null:                               ;
44592 00004680 3C20                         cmp     al,' '                  ; control code?
44593 00004682 720F                         jb  short prompt_user       ; yes, assume end of line
44594 00004684 7505                         jne short non_space             ;
44595                                       ; 10/09/2023
44596 00004686 263804                   cmp     [es:si],al ; 20h
44597                                   ;cmp   byte [es:si],' '       ;
44598 00004689 7208                       jb  short prompt_user       ;
44599                                 non_space:                              ;
44600 0000468B 88C2                         mov     dl,al                   ;
44601 0000468D B402                         mov     ah,STD_CON_OUTPUT ; 2   ;
44602 0000468F CD21                         int     21h                     ;
44603 00004691 EBE5                         jmp short line_print        ;
44604
44605                                 prompt_user:                            ;
```

```
44606 00004693 BA[1353]                       mov     dx,_$InterPrmpt          ;
44607
44608                                 generic_prompt:
44609 00004696 E8BB03                         call    print                    ;
44610                                 input_loop:                              ;
44611 00004699 B400                           mov     ah,0                     ; read a key
44612 0000469B CD16                           int     16h                      ;
44613 0000469D 08C0                           or      al,al                    ; is it a function key?
44614 0000469F 750F                           jnz short not_func         ; no
44615 000046A1 80FC3F                         cmp     ah,3Fh                   ; F5 function key?
44616 000046A4 75F3                           jne short input_loop       ; no
44617 000046A6 A0[2353]                       mov     al,[_$NO]                  ;
44618 000046A9 800E[854C]04                   or      byte [bQueryOpt],4       ; no more queries
44619 000046AE EB21                           jmp     short legal_char         ;
44620                                 not_func:
44621 000046B0 24DF                           and     al,~20h ; 0DFh           ; converting to upper case
44622 000046B2 3A06[2353]                     cmp     al,[_$NO]          ; verify character is legal
44623 000046B6 7419                           je  short legal_char       ;
44624 000046B8 3A06[1F53]                     cmp     al,[_$YES]         ;
44625 000046BC 7413                           je  short legal_char       ;
44626 000046BE 803E[6419]00                   cmp     byte [config_cmd],0      ;
44627 000046C3 74D4                           je  short input_loop       ; don't allow Esc on this query
44628 000046C5 3C1B                           cmp     al,1Bh                   ; Esc?
44629 000046C7 75D0                           jne short input_loop       ;
44630 000046C9 800E[854C]02                   or      byte [bQueryOpt],2       ; no more interactive boot prompts
44631 000046CE A0[1F53]                       mov     al,[_$YES]
44632                                 legal_char:                              ;
44633 000046D1 E888FE                         call    disp_input               ;
44634 000046D4 5E                             pop     si                       ; restore pointer to rest of CONFIG.SYS line
44635
44636 000046D5 3A06[2353]                     cmp     al,[_$NO]          ; process line?
44637 000046D9 7403                           je  short skip_cmd         ; no
44638                                     ; 12/12/2022
44639 000046DB F8                           clc
44640                                 do_cmd:
44641 000046DC 58                           pop     ax                       ;
44642                                     ; 12/12/2022
44643                                     ; cf=0
44644                                     ;clc                             ; just do the command
44645 000046DD C3                           retn
44646
44647                                 skip_cmd:
44648 000046DE 58                           pop     ax                       ;
44649                                 skip_all:
44650 000046DF B430                         mov     ah,CONFIG_REM ; '0'   ; fake out the rest of sysinit's processing
44651 000046E1 F9                           stc
44652 000046E2 C3                           retn
44653
44654                                 ;--------------------------------------------------------------------------
44655                                 ;
44656                                 ;   print_error: displays multi-config error conditions
44657                                 ;
44658                                 ;   INPUT
44659                                 ;    Carry set to pause, clear to not
44660                                 ;    ES:SI -> current command line in config.sys
44661                                 ;
44662                                 ;   OUTPUT
44663                                 ;       None
44664                                 ;
44665                                 ;   OTHER REGS USED
44666                                 ;       None
44667                                 ;
44668                                 ;   NOTES
44669                                 ;       None
44670                                 ;
44671                                 ;   HISTORY
44672                                 ;       Created 16-Mar-1992 by JeffPar
44673                                 ;
44674                                 ;--------------------------------------------------------------------------
44675
44676                                 print_error:
44677 000046E3 50                           push    ax
44678 000046E4 53                           push    bx
44679 000046E5 51                           push    cx
44680 000046E6 52                           push    dx
44681 000046E7 1E                           push    ds
44682 000046E8 0E                           push    cs
44683 000046E9 1F                           pop     ds
44684 000046EA 9C                           pushf
44685 000046EB E820FC                       call    get_linenum
44686 000046EE 891E[AF02]                   mov     [linecount],bx
44687 000046F2 E8C5E7                       call    error_line
44688 000046F5 9D                           popf
44689 000046F6 7319                         jnc short pe_ret
44690 000046F8 BA[DD51]                     mov     dx,_$PauseMsg
44691 000046FB E85603                       call    print
44692 000046FE B8070C                       mov     ax,0C07h         ; flush input buffer, then wait for key
44693 00004701 CD21                         int     21h                      ; wait for a key
44694 00004703 08C0                         or      al,al                    ; extended key?
44695 00004705 7504                         jnz short pe_1             ; no
44696 00004707 B407                         mov     ah,07h                   ; yes
44697 00004709 CD21                         int     21h                      ; eat it too
44698                                 pe_1:
44699 0000470B BA[7050]                     mov     dx,crlfm
44700 0000470E E84303                       call    print
44701                                 pe_ret:
44702 00004711 1F                           pop     ds
44703 00004712 5A                           pop     dx
44704 00004713 59                           pop     cx
44705 00004714 5B                           pop     bx
44706 00004715 58                           pop     ax
44707 00004716 C3                           retn
44708
44709                                 ;--------------------------------------------------------------------------
44710
44711                                 ;   This function is very simple: it merely prepends a "/D" to the
44712                                 ;   command-line for the shell; this (undocumented) switch disables
44713                                 ;   AUTOEXEC.BAT processing and the date/time prompt that is usually
44714                                 ;   displayed when there's no AUTOEXEC.BAT.
44715
44716                                 disable_autoexec:
44717                                     ; MSDOS 6.21 IO.SYS -  SYSINIT:4BE2h
44718                                     ; 17/04/2019 - Retro DOS v4.0
44719
44720 00004717 F606[854C]04                  test    byte [bQueryOpt],4
44721 0000471C 7443                         jz      short disable_exit
44722 0000471E F606[7B4C]01                 test    byte [dae_flag],1
44723 00004723 753C                         jnz     short disable_exit
44724 00004725 800E[7B4C]01                 or      byte [dae_flag],1
44725                                           ;or byte [bQueryOpt],2 ; MSDOS 6.0
44726 0000472A 810E[854C]0201               or       word [bQueryOpt],102h ; [bDefBlock] = 1
44727 00004730 BA4420                       mov      dx,'D ' ; 2044h
44728                                 dae_1:
44729                                     ; 03/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
```

```
44730 00004733 A0[BA4B]                       mov al,[def_swchr]
44731                                     ;mov    al,[command_line-1]     ; get default switchchar
44732 00004736 08C0                           or      al,al                   ; anything there?
44733 00004738 7427                           jz  short disable_exit    ; no, disable_autoexec already called
44734 0000473A 8A1E[BB4B]                     mov     bl,[command_line]       ;
44735 0000473E B700                           mov     bh,0                    ; BX == command-line length
44736 00004740 89D9                           mov     cx,bx                   ;
44737 00004742 80C303                         add     bl,3                    ;
44738 00004745 80FB7E                         cmp     bl,126                  ;
44739 00004748 7717                           ja  short disable_exit    ;
44740 0000474A 881E[BB4B]                     mov     [command_line],bl       ; update length
44741 0000474E 81C3[BC4B]                     add     bx,command_line+1 ; make sure we move the NULL too
44742 00004752 41                             inc     cx                      ; (just for consistency sake)
44743                                 disable_loop:                           ;
44744 00004753 8A67FD                         mov     ah,[bx-3]               ;
44745 00004756 8827                           mov     [bx],ah                 ;
44746 00004758 4B                             dec     bx                      ;
44747 00004759 E2F8                           loop    disable_loop            ;
44748 0000475B 8847FE                         mov     [bx-2],al               ;
44749                                     ;mov    word [bx-1],'D ' ; 2044h ; /D is stuffed into place now
44750 0000475E 8957FF                     mov     [bx-1],dx  ; MSDOS 6.21 IO.SYS - SYSINIT:4C29h
44751                                         ;mov       byte [command_line-1],0 ;
44752                                 disable_exit:                           ;
44753 00004761 C3                             retn
44754
44755                                 CheckQueryOpt:     ; MSDOS 6.21 IO.YSYS - SYSINIT:4C2Dh
44756 00004762 803E[854C]01               cmp     byte [bQueryOpt],1
44757 00004767 75F8                       jnz     short disable_exit
44758 00004769 F606[7B4C]02               test    byte [dae_flag],2
44759 0000476E 75F1                       jnz     short disable_exit
44760 00004770 800E[7B4C]02               or      byte [dae_flag],2
44761                                     ;mov    dx,' Y' ; (MASM syntax) ; 2059h
44762                                     ; 10/09/2023 (BugFix)
44763 00004775 BA5920                     mov     dx,'Y ' ; (NASM syntax) ; 2059h
44764 00004778 EBB9                       jmp     short dae_1
44765
44766                                 ;endif  ;MULTI_CONFIG
44767
44768                                 ;%endif     ; 02/11/2022
44769
44770
44771                                 ; 19/04/2019 - Retro DOS v4.0
44772
44773                                 ;--------------------------------------------------------------------------
44774                                 ;
44775                                 ; procedure : delim
44776                                 ;
44777                                 ;--------------------------------------------------------------------------
44778
44779                                 ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
44780                                 ; (SYSINIT:4C45h)
44781
44782                                 ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
44783                                 ;%if 0
44784                                 ;;ifdef     MULTI_CONFIG
44785                                 ;
44786                                 any_delim:
44787 0000477A 3C0D                       cmp     al,cr
44788 0000477C 7427                       je      short delim_ret
44789 0000477E 3C0A                       cmp     al,lf
44790 00004780 7423                       je      short delim_ret
44791 00004782 3C5B                       cmp     al,'['
44792 00004784 741F                       je      short delim_ret
44793 00004786 3C5D                       cmp     al,']'
44794 00004788 741B                       je      short delim_ret
44795                                 ;
44796                                 ;;endif ;MULTI_CONFIG
44797                                 ;%endif ; 02/11/2022
44798
44799                                     ; 02/11/2022 - Retrodos v4.0 (Modified MSDOS 5.0 IO.SYS)
44800                                     ; (SYSINIT:3450h)
44801                                 delim:
44802 0000478A 3C2F                       cmp     al,'/'          ; ibm will assume "/" as an delimeter.
44803 0000478C 7417                       je      short delim_ret
44804
44805 0000478E 3C00                       cmp     al,0            ; special case for sysinit!!!
44806 00004790 7413                       je      short delim_ret
44807
44808                                 org_delim:                  ; used by organize routine except for getting
44809 00004792 3C20                       cmp     al,' '           ; the filename.
44810 00004794 740F                       je      short delim_ret
44811 00004796 3C09                           cmp     al,tab ; 9
44812 00004798 740B                       je      short delim_ret
44813 0000479A 3C3D                       cmp     al,'='
44814 0000479C 7407                       je      short delim_ret
44815 0000479E 3C2C                       cmp     al,','
44816 000047A0 7403                       je      short delim_ret
44817 000047A2 3C3B                       cmp     al,';'
44818
44819                                     ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
44820
44821                                 ; 04/01/2023 - Retro DOS v4.2
44822                                 ;ifdef      MULTI_CONFIG
44823                                 ;   Make sure there's no chance of a false EOF indication
44824 000047A4 F8                         clc
44825                                 ;endif
44826                                     ; 02/11/2022
44827                                 delim_ret:
44828                                     ; 04/01/2023
44829                                     ; cf = 0
44830                                 nl_ret:     ; 10/09/2023
44831 000047A5 C3                         retn
44832
44833                                 ;--------------------------------------------------------------------------
44834                                 ;
44835                                 ; procedure : newline
44836                                 ;
44837                                 ;  newline returns with first character of next line
44838                                 ;
44839                                 ;--------------------------------------------------------------------------
44840
44841                                 newline:
44842 000047A6 E80600                     call    getchr                  ;skip non-control characters
44843 000047A9 72FA                       jc      short nl_ret
44844 000047AB 3C0A                       cmp     al,lf                   ;look for line feed
44845 000047AD 75F7                       jne     short newline
44846
44847                                     ; 10/09/2023 - Retro DOS v4.2 IO.SYS (Optimization)
44848                                     ;call   getchr
44849                                 ;nl_ret:
44850                                     ;retn
44851                                     ; 10/09/2023
44852                                     ;jmp    short getchr
44853
```

```
                                ; 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]
```

```
44978 000047F6 A1[6203]                 mov     ax,[memlo]
44979
44980 000047F9 E819CB                   call    ParaRound               ; para round up
44981
44982                                   ;add    [cs:memhi],ax
44983 000047FC 0106[6403]               add     [memhi],ax
44984                                   ;mov    word [cs:memlo],0
44985 00004800 C706[6203]0000           mov     word [memlo],0
44986                                   ;mov    ax,[cs:memhi]           ; ax = new memhi
44987 00004806 A1[6403]                 mov     ax,[memhi]
44988                                   ;cmp    ax,[cs:ALLOCLIM]        ; if new memhi >= alloclim, error
44989 00004809 3B06[A502]               cmp     ax,[ALLOCLIM]
44990                                   ;jae    short mem_err
44991                                   ; 13/04/2024
44992 0000480D 7322                     jae     short mem_err2 ; ds = cs
44993                                   ;test   byte [cs:setdevmarkflag],for_devmark ; 2
44994 0000480F F606[6919]02             test    byte [setdevmarkflag],for_devmark ; 2
44995 00004814 7416                     jz      short skip_set_devmarksize
44996 00004816 06                       push    es
44997 00004817 56                       push    si
44998                                   ;mov    si,[cs:devmark_addr]
44999 00004818 8B36[6719]               mov     si,[devmark_addr]
45000 0000481C 8EC6                     mov     es,si
45001 0000481E 29F0                     sub     ax,si
45002 00004820 48                       dec     ax
45003                                   ;mov    [es:3],ax
45004 00004821 26A30300                 mov     [es:devmark.size],ax  ; paragraph
45005                                   ;and    byte [cs:setdevmarkflag],not_for_devmark ; 0FDh
45006 00004825 8026[6919]FD             and     byte [setdevmarkflag],not_for_devmark ; 0FDh
45007 0000482A 5E                       pop     si
45008 0000482B 07                       pop     es
45009                                 skip_set_devmarksize:
45010 0000482C 58                       pop     ax
45011
45012                                   ; 10/09/2023
45013 0000482D 1F                       pop     ds
45014
45015                                   ; 11/12/2022
45016                                   ; cf = 0
45017                                   ; 02/11/2022
45018                                   ;clc    ; ? (not needed here) ; clear carry
45019 0000482E C3                       retn
45020
45021                                 ;--------------------------------------------------------------------------
45022
45023                                 mem_err:
45024                                   ; 11/12/2022
45025 0000482F 0E                       push    cs
45026 00004830 1F                       pop     ds
45027                                 mem_err2:
45028 00004831 BA[4951]                 mov     dx,badmem
45029                                   ;push   cs
45030                                   ;pop    ds
45031 00004834 E81D02                   call    print
45032 00004837 E914CB                   jmp     stall
45033
45034                                 ;--------------------------------------------------------------------------
45035                                 ;
45036                                 ; procedure : calldev
45037                                 ;
45038                                 ;--------------------------------------------------------------------------
45039
45040                                   ; 02/11/2022 - Retrodos v4.0 (Modified MSDOS 5.0 IO.SYS)
45041                                   ; (SYSINIT:34E0h)
45042
45043                                   ; 13/04/2024 - Retrodos v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
45044                                   ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:4F3Eh)
45045
45046                                 calldev:
45047 0000483A 2E8E1E[3B24]             mov     ds,[cs:DevEntry+2]
45048 0000483F 2E031E[3924]             add     bx,[cs:DevEntry]      ; do a little relocation
45049 00004844 8B07                     mov     ax,[bx]
45050
45051 00004846 2EFF36[3924]              push    word [cs:DevEntry]
45052 0000484B 2EA3[3924]               mov     [cs:DevEntry],ax
45053 0000484F BB[6F03]                 mov     bx,packet
45054 00004852 2EFF1E[3924]             call    far [cs:DevEntry]
45055 00004857 2E8F06[3924]             pop     word [cs:DevEntry]
45056 0000485C C3                       retn
45057
45058                                 ;--------------------------------------------------------------------------
45059                                 ;
45060                                 ; procedure : todigit
45061                                 ;
45062                                 ;--------------------------------------------------------------------------
45063
45064                                 todigit:
45065 0000485D 2C30                     sub     al,'0'
45066                                   ;jb     short notdig  ; 02/11/2022
45067                                   ; 12/12/2022
45068 0000485F 7203                     jb      short notdig2
45069                                   ;cmp    al,9
45070                                   ;ja     short notdig
45071                                   ;clc
45072                                   ;retn
45073                                   ; 12/12/2022
45074 00004861 3C0A                     cmp     al,10
45075 00004863 F5                       cmc
45076                                 notdig:
45077                                   ;stc
45078                                 notdig2:
45079 00004864 C3                       retn
45080
45081                                 ;--------------------------------------------------------------------------
45082                                 ;
45083                                 ; procedure : getnum
45084                                 ;
45085                                 ; getnum parses a decimal number.
45086                                 ; returns it in ax, sets zero flag if ax = 0 (may be considered an
45087                                 ; error), if number is bad carry is set, zero is set, ax=0.
45088                                 ;
45089                                 ;--------------------------------------------------------------------------
45090
45091                                 getnum:
45092 00004865 53                       push    bx
45093 00004866 31DB                     xor     bx,bx                 ; running count is zero
45094                                 b2:
45095 00004868 E8F2FF                   call    todigit               ; do we have a digit ?
45096 0000486B 7247                     jc      short badnum          ; no, bomb
45097
45098 0000486D 93                       xchg    ax,bx                 ; put total in ax
45099 0000486E 53                       push    bx                    ; save digit (0 to 9)
45100                                   ;mov    bx,10                 ; base of arithmetic
45101                                   ; 12/12/2022
```

```
45102 0000486F B30A                     mov     bl,10
45103 00004871 F7E3                     mul     bx                      ; shift by one decimal digit
45104 00004873 5B                       pop     bx                      ; get back digit (0 to 9)
45105 00004874 00D8                     add     al,bl                   ; get total
45106 00004876 80D400                   adc     ah,0                    ; make that 16 bits
45107 00004879 7239                     jc      short badnum            ; too big a number
45108
45109 0000487B 93                       xchg    ax,bx                   ; stash total
45110
45111 0000487C E830FF                   call    getchr                  ;get next digit
45112 0000487F 722D                     jc      short b1                ; no more characters
45113 00004881 3C20                     cmp     al,' '                  ; space?
45114 00004883 741F                     je      short b15               ; then end of digits
45115 00004885 3C2C                     cmp     al,','                  ; ',' is a seperator!!!
45116 00004887 741B                     je      short b15               ; then end of digits.
45117 00004889 3C09                     cmp     al, tab ; 9             ; tab
45118 0000488B 7417                     je      short b15
45119 0000488D 2E3A06[AE02]             cmp     al,[cs:sepchr]          ; allow 0 or special separators
45120 00004892 7410                     je      short b15
45121 00004894 3C2F                     cmp     al,'/'                  ; see if another switch follows
45122                                   ; 12/12/2022
45123                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
45124                                   ;nop                            ; cas - remnant of old bad code
45125                                   ;nop
45126 00004896 740C                     je      short b15
45127 00004898 3C0A                     cmp     al,lf                   ; line-feed?
45128 0000489A 7408                     je      short b15
45129 0000489C 3C0D                     cmp     al,cr                   ; carriage return?
45130 0000489E 7404                     je      short b15
45131 000048A0 08C0                     or      al,al                   ; end of line separator?
45132 000048A2 75C4                     jnz     short b2                ; no, try as a valid char...
45133                                 b15:
45134 000048A4 2EFF06[5603]             inc     word [cs:count]                 ; one more character to s...
45135 000048A9 2EFF0E[5A03]             dec     word [cs:chrptr]        ; back up over separator
45136                                 b1:
45137 000048AE 89D8                     mov     ax,bx                   ; get proper count
45138 000048B0 09C0                     or      ax,ax                   ; clears carry, sets zero accordingly
45139 000048B2 5B                       pop     bx
45140 000048B3 C3                       retn
45141                                 badnum:
45142                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
45143                                   ;mov    byte [cs:sepchr],0
45144 000048B4 31C0                     xor     ax,ax                   ; set zero flag, and ax = 0
45145                                   ; 12 /12/2022
45146 000048B6 2EA2[AE02]               mov     [cs:sepchr],al ; 0
45147 000048BA 5B                       pop     bx
45148 000048BB F9                       stc                             ; and carry set
45149 000048BC C3                       retn
45150
45151                                 ;******************************************************************************
45152
45153                                 setdoscountryinfo:
45154
45155                                 ;------------------------------------------------------------------------------
45156                                 ;input: es:di -> pointer to dos_country_cdpg_info
45157                                 ;   ds:0  -> buffer.
45158                                 ;   si = 0
45159                                 ;   ax = country id
45160                                 ;   dx = code page id. (if 0, then use ccsyscodepage as a default.)
45161                                 ;   bx = file handle
45162                                 ;   this routine can handle maximum 438 country_data entries.
45163                                 ;
45164                                 ;output: dos_country_cdpg_info set.
45165                                 ;    carry set if any file read failure or wrong information in the file.
45166                                 ;    carry set and cx = -1 if cannot find the matching country_id,
45167                                 ;    codepage_id in the file.
45168                                 ;------------------------------------------------------------------------------
45169
45170                                   ; 04/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
45171                                   ; (SYSINIT:4D83h)
45172
45173                                   ; 10/09/2023 - Retrodos v4.2 IO.SYS (Optimization)
45174                                   ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:4FCAh)
45175
45176 000048BD 57                       push    di
45177 000048BE 50                       push    ax
45178 000048BF 52                       push    dx
45179
45180 000048C0 31C9                     xor     cx,cx
45181 000048C2 31D2                     xor     dx,dx
45182 000048C4 B80002                   mov     ax,512                  ;read 512 bytes
45183 000048C7 E84301                   call    readincontrolbuffer     ;read the file header
45184 000048CA 724A                     jc      short setdosdata_fail
45185
45186 000048CC 06                       push    es
45187 000048CD 56                       push    si
45188
45189 000048CE 0E                       push    cs
45190 000048CF 07                       pop     es
45191
45192 000048D0 BF[204B]                 mov     di,country_file_signature ; db 0FFh,'COUNTRY'
45193 000048D3 B90800                   mov     cx,8                    ;length of the signature
45194 000048D6 F3A6                     repz    cmpsb
45195
45196 000048D8 5E                       pop     si
45197 000048D9 07                       pop     es
45198 000048DA 753A                     jnz     short setdosdata_fail ;signature mismatch
45199
45200 000048DC 83C612                   add     si,18                   ;si -> county info type
45201 000048DF 803C01                   cmp     byte [si],1             ;only accept type 1 (currently only 1 header type)
45202 000048E2 7532                     jne     short setdosdata_fail ;cannot proceed. error return
45203
45204 000048E4 46                       inc     si                      ;si -> file offset
45205 000048E5 8B14                     mov     dx,[si]                 ;get the info file offset.
45206 000048E7 8B4C02                   mov     cx,[si+2]
45207 000048EA B80018                   mov     ax,6144                 ;read 6144 bytes.
45208 000048ED E81D01                   call    readincontrolbuffer     ;read info
45209 000048F0 7224                     jc      short setdosdata_fail
45210
45211 000048F2 8B0C                     mov     cx,[si]                 ;get the # of country, codepage combination entries
45212 000048F4 81F9B601                 cmp     cx,438                  ;cannot handle more than 438 entries.
45213 000048F8 771C                     ja      short setdosdata_fail
45214
45215 000048FA 46                       inc     si
45216 000048FB 46                       inc     si                      ;si -> entry information packet
45217 000048FC 5A                       pop     dx                      ;restore code page id
45218 000048FD 58                       pop     ax                      ;restore country id
45219 000048FE 5F                       pop     di
45220
45221                                 setdoscntry_find:               ;search for desired country_id,codepage_id.
45222 000048FF 3B4402                   cmp     ax,[si+2]               ;compare country_id
45223 00004902 7509                     jne     short setdoscntry_next
45224
45225                                   ;cmp    dx,0                    ;no user specified code page ?
```

```
45226                                   ;je     short setdoscntry_any_codepage ;then no need to match code page id.
45227                                   ; 10/09/2023
45228 00004904 09D2                     or      dx,dx ; cmp dx,0
45229 00004906 7413                     jz      short setdoscntry_any_codepage
45230 00004908 3B5404                   cmp     dx,[si+4]               ;compare code page id
45231 0000490B 7411                     je      short setdoscntry_got_it
45232
45233                                 setdoscntry_next:
45234 0000490D 0334                     add     si,[si]                 ;next entry
45235 0000490F 46                       inc     si
45236 00004910 46                       inc     si                      ;take a word for size of entry itself
45237 00004911 E2EC                     loop    setdoscntry_find
45238
45239                                   ;mov    cx,-1                   ;signals that bad country id entered.
45240                                   ; 10/09/2023
45241 00004913 49                       dec     cx ; 0 -> -1
45242                                 setdoscntry_fail:
45243 00004914 F9                       stc
45244 00004915 C3                       retn
45245
45246                                 setdosdata_fail:
45247 00004916 5E                       pop     si
45248 00004917 59                       pop     cx
45249 00004918 5F                       pop     di
45250 00004919 EBF9                     jmp     short setdoscntry_fail
45251
45252                                 setdoscntry_any_codepage:       ;use the code_page_id of the country_id found.
45253 0000491B 8B5404                   mov     dx,[si+4]
45254
45255                                 setdoscntry_got_it:                     ;found the matching entry
45256 0000491E 2E8916[284B]             mov     [cs:cntrycodepage_id],dx ;save code page id for this country.
45257 00004923 8B540A                   mov     dx,[si+10]              ;get the file offset of country data
45258 00004926 8B4C0C                   mov     cx,[si+12]
45259 00004929 B80002                   mov     ax,512                  ;read 512 bytes
45260 0000492C E8DE00                   call    readincontrolbuffer
45261 0000492F 72E3                     jc      short setdoscntry_fail
45262
45263 00004931 8B0C                     mov     cx,[si]                 ;get the number of entries to handle.
45264 00004933 46                       inc     si
45265 00004934 46                       inc     si                      ;si -> first entry
45266
45267                                 setdoscntry_data:
45268 00004935 57                       push    di                      ;es:di -> dos_country_cdpg_info
45269 00004936 51                       push    cx                      ;save # of entry left
45270 00004937 56                       push    si                      ;si -> current entry in control buffer
45271
45272 00004938 8A4402                   mov     al,[si+2]               ;get data entry id
45273 0000493B E8A400                   call    getcountrydestination ;get the address of destination in es:di
45274 0000493E 727C                     jc      short setdoscntry_data_next ;no matching data entry id in dos
45275
45276 00004940 8B5404                   mov     dx,[si+4]               ;get offset of data
45277 00004943 8B4C06                   mov     cx,[si+6]
45278 00004946 B80042                   mov     ax,4200h
45279 00004949 F9                       stc
45280 0000494A CD21                     int     21h                     ;move pointer
45281 0000494C 72C8                     jc      short setdosdata_fail
45282
45283 0000494E BA0002                   mov     dx,512                  ;start of data buffer
45284 00004951 B91400                   mov     cx,20                   ;read 20 bytes only. we only need to
45285 00004954 B43F                     mov     ah,3Fh                  ;look at the length of the data in the file.
45286 00004956 F9                       stc
45287 00004957 CD21                     int     21h                     ;read the country.sys data
45288 00004959 72BB                     jc      short setdosdata_fail ;read failure
45289
45290 0000495B 39C8                     cmp     ax,cx
45291 0000495D 75B7                     jne     short setdosdata_fail ; 13/05/2019
45292
45293 0000495F 8B5404                   mov     dx,[si+4]               ;get offset of data again.
45294 00004962 8B4C06                   mov     cx,[si+6]
45295 00004965 B80042                   mov     ax,4200h
45296 00004968 F9                       stc
45297 00004969 CD21                     int     21h                     ;move pointer back again
45298 0000496B 72A9                     jc      short setdosdata_fail
45299
45300 0000496D 56                       push    si
45301 0000496E BE0802                   mov     si,(512+8)              ;get length of the data from the file
45302 00004971 8B0C                     mov     cx,[si]
45303 00004973 5E                       pop     si
45304 00004974 BA0002                   mov     dx,512                  ;start of data buffer
45305 00004977 83C10A                   add     cx,10                   ;signature + a word for the length itself
45306 0000497A B43F                     mov     ah,3Fh                  ;read the data from the file.
45307 0000497C F9                       stc
45308 0000497D CD21                     int     21h
45309 0000497F 7295                     jc      short setdosdata_fail
45310
45311 00004981 39C8                     cmp     ax,cx
45312 00004983 7591                     jne     short setdosdata_fail
45313
45314 00004985 8A4402                   mov     al,[si+2]               ;save data id for future use.
45315 00004988 BE0802                   mov     si,(512+8)              ;si-> data buffer + id tag field
45316 0000498B 8B0C                     mov     cx,[si]                 ;get the length of the file
45317 0000498D 41                       inc     cx                      ;take care of a word for lenght of tab
45318 0000498E 41                       inc     cx                      ;itself.
45319 0000498F 81F9F805                 cmp     cx,(2048-512-8)         ; 1528 ;fit into the buffer?
45320 00004993 7781                     ja      short setdosdata_fail
45321
45322                                   ;if     bugfix
45323 00004995 E83100                   call    setdbcs_before_copy
45324                                   ;endif
45325
45326 00004998 3C01                      cmp     al,SetCountryInfo ; 1 ;is the data for setcountryinfo table?
45327 0000499A 7511                     jne     short setdoscntry_mov ;no, don't worry
45328
45329 0000499C 26FF7518                  push    word [es:di+country_cdpg_info.ccMono_Ptr-country_cdpg_info.ccCountryInfoLen]
45330                                   ;push   word [es:di+24]                 ;cannot destroy ccmono_ptr address. save them.
45331 000049A0 26FF751A                 push    word [es:di+country_cdpg_info.ccMono_Ptr-country_cdpg_info.ccCountryInfoLen+2]
45332                                   ;push   word [es:di+26]                 ;at this time di -> cccountryinfolen
45333
45334 000049A4 57                       push    di                      ;save di
45335
45336                                   ;push   ax
45337                                   ;mov    ax,[cs:cntrycodepage_id] ;do not use the code page info in country_info
45338                                   ;mov    [si+4],ax               ;use the saved one for this !!!!
45339                                   ;pop    ax
45340                                   ; 10/09/2023
45341 000049A5 2EFF36[284B]             push    word [cs:cntrycodepage_id]
45342 000049AA 8F4404                   pop     word [si+4]
45343
45344                                 setdoscntry_mov:
45345 000049AD F3A4                     rep     movsb                   ;copy the table into dos
45346 000049AF 3C01                     cmp     al,SetCountryInfo       ;was the ccmono_ptr saved?
45347 000049B1 7509                     jne     short setdoscntry_data_next
45348
45349 000049B3 5F                       pop     di                      ;restore di
```

```
45350 000049B4 268F451A                 pop     word [es:di+country_cdpg_info.ccMono_Ptr-country_cdpg_info.ccCountryInfoLen+2]
45351                                   ;pop    word [es:di+26]             ;restore
45352 000049B8 268F4518                 pop     word [es:di+country_cdpg_info.ccMono_Ptr-country_cdpg_info.ccCountryInfoLen]
45353                                   ;pop    word [es:di+24]
45354
45355                                 setdoscntry_data_next:
45356 000049BC 5E                       pop     si                  ;restore control buffer pointer
45357 000049BD 59                       pop     cx                  ;restore # of entries left
45358 000049BE 5F                       pop     di                  ;restore pointer to dso_country_cdpg
45359 000049BF 0334                     add     si,[si]             ;try to get the next entry
45360 000049C1 46                       inc     si
45361 000049C2 46                       inc     si                  ;take a word of entry length itself
45362 000049C3 49                       dec     cx
45363                                   ; 10/09/2023
45364 000049C4 741B                     jz      short setdoscntry_ok
45365                                   ;cmp    cx,0
45366                                   ;je     short setdoscntry_ok
45367 000049C6 E96CFF                   jmp     setdoscntry_data
45368
45369                                   ; 18/12/2022
45370                                 ;setdoscntry_ok:
45371                                   ;retn
45372
45373                                 ;--------------------------------------------------------------------------
45374
45375                                   ;if     bugfix
45376
45377                                   ; 10/09/2023 - Retrodos v4.2 IO.SYS (Optimization)
45378
45379                                 setdbcs_before_copy:
45380 000049C9 3C07                     cmp     al,SetDBCS ; 7      ; dbcs vector set?
45381 000049CB 7514                     jne     short sdbcsbc       ; jump if not
45382
45383                                   ; 10/09/2023
45384 000049CD 50                       push    ax
45385 000049CE 31C0                     xor     ax,ax
45386 000049D0 263905                   cmp     [es:di],ax ; 0
45387 000049D3 740B                     je      short sdbcsbc_pop
45388
45389                                   ;cmp    word [es:di],0      ; zero byte data block?
45390                                   ;je     short sdbcsbc       ; jump if so
45391
45392 000049D5 57                       push    di
45393                                   ; 10/09/2023
45394                                   ;push   ax
45395 000049D6 51                       push    cx
45396 000049D7 268B0D                   mov     cx,[es:di]          ; load block length
45397                                   ;add    di,2                ; points actual data
45398 000049DA 47                       inc     di
45399 000049DB 47                       inc     di
45400                                   ;xor    al,al               ; fill bytes
45401 000049DC F3AA                     rep     stosb               ; clear data block
45402 000049DE 59                       pop     cx
45403                                   ;pop    ax
45404 000049DF 5F                       pop     di
45405
45406                                 sdbcsbc_pop:        ; 10/09/2023
45407 000049E0 58                       pop     ax
45408                                 sdbcsbc:
45409                                 setdoscntry_ok:     ; 18/12/2022
45410 000049E1 C3                       retn
45411
45412                                   ;endif
45413
45414                                 ;--------------------------------------------------------------------------
45415
45416                                 getcountrydestination:
45417
45418                                 ;--------------------------------------------------------------------------
45419                                 ;get the destination address in the dos country info table.
45420                                 ;
45421                                 ;input: al - data id
45422                                 ;   es:di -> dos_country_cdpg_info
45423                                 ;on return:
45424                                 ;   es:di -> destination address of the matching data id
45425                                 ;   carry set if no matching data id found in dos.
45426                                 ;--------------------------------------------------------------------------
45427
45428                                   ; 04/01/2023 - Retrodos v4.2 (Modified MSDOS 6.21 IO.SYS)
45429                                   ; (SYSINIT:4EB2h)
45430
45431                                   ; 10/09/2023 - Retrodos v4.2 IO.SYS (Optimization)
45432                                   ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:50F9h)
45433
45434 000049E2 51                       push    cx
45435                                   ;add    di,74
45436 000049E3 83C74A                   add     di,country_cdpg_info.ccNumber_of_entries
45437                                                               ;skip the reserved area, syscodepage etc.
45438 000049E6 268B0D                   mov     cx,[es:di]          ;get the number of entries
45439 000049E9 47                       inc     di
45440 000049EA 47                       inc     di                  ;si -> the first start entry id
45441
45442                                 getcntrydest:
45443 000049EB 263805                   cmp     byte [es:di],al
45444 000049EE 7413                     je      short getcntrydest_ok
45445
45446 000049F0 26803D01                 cmp     byte [es:di],SetCountryInfo ;was it setcountryinfo entry?
45447 000049F4 7405                     je      short getcntrydest_1
45448
45449 000049F6 83C705                   add     di,5                ;next data id
45450 000049F9 EB03                     jmp     short getcntrydest_loop
45451
45452                                 getcntrydest_1:
45453                                   ;add    di,41
45454 000049FB 83C729                   add     di,NEW_COUNTRY_SIZE+3 ;next data id
45455                                 getcntrydest_loop:
45456 000049FE E2EB                     loop    getcntrydest
45457 00004A00 F9                       stc
45458                                   ;jmp    short getcntrydest_exit
45459                                 getcntrydest_exit:
45460                                   ; 10/09/2023
45461 00004A01 59                       pop     cx
45462 00004A02 C3                       retn
45463
45464                                 getcntrydest_ok:
45465                                   ; 10/09/2023
45466 00004A03 47                       inc     di
45467
45468                                 ;   cmp     al,SetCountryInfo ; 1 ;select country info?
45469                                 ;   jne     short getcntrydest_ok1
45470                                 ;
45471                                 ;   ;inc    di                  ;now di -> cccountryinfolen
45472                                 ;   jmp     short getcntrydest_exit
45473
```

```
                                    ; 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
                                    ;;mov   ah,IOCTL ; 44h
                                    ;mov    ax,(IOCTL<<8) ; 13/05/2019
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                    ;xor    ax,ax
                                    ;mov    ah,44h  ; IOCTL
                                    ; 11/12/2022
00004A69 B80044                     mov     ax,4400h ; IOCTL<<8

00004A6C CD21                       int     21h

00004A6E F6C280                     test    dl,10000000b ; 80h
00004A71 75F3                       jnz     short of_retn

00004A73 B43E                       mov     ah,CLOSE ; 3Eh
00004A75 CD21                       int     21h
00004A77 EBE5                       jmp     short open_dev1

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

                                ; 18/12/2022
                                %if 0
                                open_file:
                                    mov     ah,OPEN ; 3Dh
                                    stc
                                    int     21h
                                    retn
                                %endif

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

                                ; test int24. return back to dos with the fake user response of "fail"

                                int24:
00004A79 B003                       mov     al,3                    ; fail the system call
00004A7B CF                         iret                            ; return back to dos.

                                ; 19/04/2019 - Retro DOS v4.0

                                ;--------------------------------------------------------------------------
                                ; DATA
                                ;--------------------------------------------------------------------------

                                ;include copyrigh.inc                   ; copyright statement

                                ; MSDOS 6.21 IO.SYS - SYSINIT:4FA3h

                                ;MsDosVersion6Copyr:
                                ;   db      'MS DOS Version 6 (C)Copyright 1981-1993 Microsoft Corp '
                                ;   db      'Licensed Material - Property of Microsoft All rights reserved '

                                ; 22/10/2022
                                ; MSDOS 5.0 IO.SYS - SYSINIT:378Ch

                                ; 28/12/2022
                                %if 0
                                    ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                MsDosVersion5Copyr:
                                    db      'MS DOS Version 5.00 (C)Copyright 1981-1991 Microsoft Corp '
                                    db      'Licensed Material - Property of Microsoft All rights reserved '
                                %endif

                                ; 13/04/2024 - Retro DOS v5.0
                                ; PCDOS 7.1 IBMBIO.COM - SYSINIT:51EAh   (IBMBIO.COM offset 42266)
                                %if 0
                                IBMDOSV71COPYR:
                                    db 'IBM DOS Version 7.1 (C)Copyright 1981-2002 IBM Corporation '
                                    db 'Licensed Material - Property of IBM All rights reserved '
                                %endif

                                ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
                                ; 22/10/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
                                ; 20/04/2019 - Retro DOS v4.0
                                ;BOOTMES:
                                ;   db      13
                                ;   db      10
                                ;   db      "MS-DOS version "
                                ;   db      MAJOR_VERSION + "0"
                                ;   db      "."
                                ;   db      (MINOR_VERSION / 10) + "0"
                                ;   db      (MINOR_VERSION % 10) + "0"
                                ;   db      13,10
                                ;   ;db    "Copyright 1981-1993 Microsoft Corp.",13,10,"$"
                                ;   ; 22/10/2022
                                ;   db      "Copyright 1981-1991 Microsoft Corp.",13,10,"$"
                                ;   ;
                                ;   db      0

                                    ; 01/01/2023 - Retro DOS v4.2

                                    ; 28/12/2022 - Retro DOS v4.1
                                ;MsDosVersion5Copyr:
                                ;   db      13,10,"MS DOS Version 5.0"
                                ;   db      13,10,"Copyright 1981-1991 Microsoft Corp.",13,10,"$",0

                                    ; 12/12/2022
00004A7C 00                         db      0
                                ; 12/12/2022
                                BOOTMES:
00004A7D 0D0A                       db      13,10
                                    ;;;db   "Retro DOS v4.0 (Modified MSDOS 5.0) "
                                    ; 28/12/2022
                                    ;;db    "Retro DOS v4.1 (Modified MSDOS 5.0) "
                                    ; 01/01/2023
                                    ;db     "Retro DOS v4.2 (Modified MSDOS 6.22) "
                                    ; 30/12/2023
00004A7F 526574726F20444F53-        db      "Retro DOS v5.0 (Modified PCDOS 7.1) "
00004A88 2076352E3020284D6F-
00004A91 646966696564205043-
00004A9A 444F5320372E312920

00004AA3 0D0A                       db      13,10
                                    ;db     "by Erdogan Tan [2024] " ; 01/01/2024
00004AA5 6279204572646F6761-        db      "by Erdogan Tan [2026] " ; 19/01/2026
00004AAE 6E2054616E205B3230-
00004AB7 32365D20
00004ABB 0D0A                       db      13,10
00004ABD 0D0A2400                   db      13,10,"$",0

00004AC1 4E554C00               nuldev:     db      "NUL",0
00004AC5 434F4E00               condev:     db      "CON",0
00004AC9 41555800               auxdev:     db      "AUX",0
00004ACD 50524E00               prndev:     db      "PRN",0
```

```
45841                                   ;IFDEF      CONFIGPROC
45842 00004AD1 5C434F4E4649472E53-      config:     db      "\CONFIG.SYS",0
45842 00004ADA 595300
45843
45844 00004ADD 413A                     cntry_drv:  db      "A:"
45845 00004ADF 5C                       cntry_root: db      "\"
45846 00004AE0 434F554E5452592E53-      cntry_path: db      "COUNTRY.SYS",0
45846 00004AE9 595300
45847                                           ;db 52 dup (0)
45848 00004AEC 00<rep 34h>                      times 52 db 0
45849
45850                                   country_file_signature:
45851 00004B20 FF434F554E545259             db      0FFh,'COUNTRY'
45852
45853                                   cntrycodepage_id:
45854 00004B28 0000                         dw      0
45855
45856                                   ;ENDIF ; CONFIGPROC
45857
45858                                   ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
45859                                   ; (SYSINIT:5081h)
45860
45861                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
45862                                   ;ifdef      MULTI_CONFIG
45863 00004B2A 00                       newcmd:  db 0                       ; non-zero if non-std shell specified
45864 00004B2B 40                       tmplate: db 64                        ; must precede commnd
45865                                   ;endif
45866
45867                                   ;ifdef ROMEXEC
45868                                   ;   db      7                         ; size of commnd line (excl. null)
45869                                   ;commnd: db "COMMAND",0
45870                                   ;   db      56 dup (0)
45871                                   ;else
45872                                       ; 02/11/2022
45873 00004B2C 0C                          db      12                        ; size of commnd line (excl. null)
45874 00004B2D 5C434F4D4D414E442E-      commnd:     db      "\COMMAND.COM",0
45874 00004B36 434F4D00
45875                                       ;db     51 dup (0)
45876 00004B3A 00<rep 33h>                  times   51 db 0
45877                                   ;endif
45878
45879                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
45880                                   ;ifdef      MULTI_CONFIG
45881 00004B6D 5C434F4D4D414E442E-      commnd2: db         "\COMMAND.COM",0      ; alternate commands to exec,
45881 00004B76 434F4D00
45882 00004B7A 022F5000                      db     2,"/P",0               ; followed by their respective alternate
45883 00004B7E 5C4D53444F535C434F-      commnd3: db "\MSDOS\COMMAND.COM",0; command lines
45883 00004B87 4D4D414E442E434F4D-
45883 00004B90 00
45884 00004B91 0B413A5C4D53444F53-           db     11,"A:\MSDOS /P",0    ;(the drive letter are dynamically replaced)
45884 00004B9A 202F5000
45885 00004B9E 5C444F535C434F4D4D-      commnd4: db "\DOS\COMMAND.COM",0  ;
45885 00004BA7 414E442E434F4D00
45886 00004BAF 09413A5C444F53202F-           db     9,"A:\DOS /P",0               ;
45886 00004BB8 5000
45887                                   def_swchr:
45888 00004BBA 00                            db     0                      ; default switchchar (referenced as command_line-1)
45889                                   ;endif
45890                                       ; 30/10/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
45891                                   command_line:
45892 00004BBB 022F50                       db      2,"/P"                 ; default command.com args
45893                                       ;db     125 dup (0)
45894 00004BBE 00<rep 7Dh>                  times   125 db 0
45895
45896                                   pathstring:
45897                                       ;db     64 dup (0)
45898 00004C3B 00<rep 40h>                  times   64 db 0
45899
45900                                   ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
45901                                   ; (SYSINIT:51D3h)
45902                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
45903                                   ;%if 0
45904
45905                                   dae_flag:
45906 00004C7B 00                          db      0 ; MSDOS 6.21 IO.SYS - SYSINIT:51D2h
45907
45908                                   ;ifdef      MULTI_CONFIG
45909
45910                                   ; 04/03/2022- Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS, SYSINIT)
45911                                   MAX_MULTI_CONFIG equ 9     ; max # of multi-config menu items supported
45912
45913                                   ;   Beware of byte pairs accessed as words (see all "KEEP AFTER" notes below)
45914
45915 00004C7C 07                       bMenuColor: db      07h ; 1Fh       ; default fgnd/bgnd color
45916 00004C7D 00                       bMenuPage:  db      0               ; menu video page (KEEP AFTER bMenuColor)
45917 00004C7E 05                                   db      5               ; video page function # (KEEP AFTER bMenuPage)
45918 00004C7F 00                       bLastCol:   db      0               ; ending column on status line
45919 00004C80 18                       bLastRow:   db      24              ; row # of status line (KEEP AFTER bLastCol)
45920 00004C81 00                       bDisableUI: db      0               ; 1=disable clean/interactive
45921                                                                           ; 2=disable default 2-second delay
45922 00004C82 00                       bCRTPage:   db      0               ; value saved from BIOS data area
45923 00004C83 0000                     wCRTStart:  dw      0               ; value saved from BIOS data area
45924 00004C85 00                       bQueryOpt:  db      0               ; 0=off, 1=prompt all, 2=prompt none, 4=skip all
45925 00004C86 01                       bDefBlock:  db      1               ; default block #
45926 00004C87 00                       bMaxBlock:  db      0               ; maxmimum block #
45927 00004C88 0000                     offDefBlock:        dw      0               ; offset of name of default block (if any)
45928 00004C8A FF                       secTimeOut: db      -1 ; 0FFh       ; # of seconds for timeout (-1 == indefinite)
45929 00004C8B 00                       secElapsed: db      0               ; # of seconds elapsed so far (KEEP AFTER secTimeOut)
45930 00004C8C 00<rep Ah>               abBlockType:        times MAX_MULTI_CONFIG+1 db 0 ; array of block types
45931 00004C96 0000<rep Ah>             aoffBlockName:      times MAX_MULTI_CONFIG+1 dw 0 ; array of offsets of block names
45932 00004CAA 0000<rep Ah>             aoffBlockDesc:      times MAX_MULTI_CONFIG+1 dw 0 ; array of offsets of block descriptions
45933
45934 00004CBE 434F4E4649473D00         szBoot:             db      "CONFIG=",0
45935 00004CC6 4D454E5500               szMenu:             db      "MENU",0
45936 00004CCB 434F4D4D4F4E00           szCommon:   db      "COMMON",0
45937
45938                                   ;endif      ;MULTI_CONFIG
45939
45940                                       ; 10/09/2023
45941                                       ; MSDOS 6.21 IO.SYS - SYSINIT:5229h
45942                                       ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:546Eh)
45943
45944                                   comtab:     ; label byte
45945
45946                                   ;           cmd len     command         cmd code
45947                                   ;           -------     -------         --------
45948
45949                                   ;ifdef MULTI_CONFIG
45950 00004CD2 015B5B                          db      1,      "[",            CONFIG_BEGIN
45951                                   ;endif
45952 00004CD5 0542452454144B43                db      5,      "BREAK",        CONFIG_BREAK
45953 00004CDC 0742552464645525342             db      7,      "BUFFERS",      CONFIG_BUFFERS
45954 00004CE5 07434F4D4D454E5459              db      7,      "COMMENT",      CONFIG_COMMENT
45955 00004CEE 07434F554E54525951              db      7,      "COUNTRY",      CONFIG_COUNTRY
```

```
45956 00004CF7 0644455649434544                 db      6,      "DEVICE",       CONFIG_DEVICE
45957 00004CFF 0A44455649434544849-             db      10,     "DEVICEHIGH",   CONFIG_DEVICEHIGH
45957 00004D08 474855
45958 00004D0B 03444F5348                       db      3,      "DOS",          CONFIG_DOS
45959 00004D10 0844524956041524D-               db      8,      "DRIVPARM",     CONFIG_DRIVPARM
45959 00004D19 50
45960 00004D1A 044643425358                     db      4,      "FCBS",         CONFIG_FCBS
45961 00004D20 0546494C455346                   db      5,      "FILES",        CONFIG_FILES
45962                                   ;ifdef MULTI_CONFIG
45963 00004D27 07494E434C5544454A               db      7,      "INCLUDE",      CONFIG_INCLUDE
45964                                   ;endif
45965 00004D30 07494E5354414C4C49               db      7,      "INSTALL",      CONFIG_INSTALL
45966 00004D39 0B494E5354414C4C48-              db      11,     "INSTALLHIGH",  CONFIG_INSTALLHIGH
45966 00004D42 49474857
45967 00004D46 094C41535444524956-              db      9,      "LASTDRIVE",    CONFIG_LASTDRIVE
45967 00004D4F 454C
45968                                   ;ifdef MULTI_CONFIG
45969 00004D51 075355424D454E554F               db      7,      "SUBMENU",      CONFIG_SUBMENU
45970 00004D5A 094D454E55434F4C4F-              db      9,      "MENUCOLOR",    CONFIG_MENUCOLOR
45970 00004D63 5252
45971 00004D65 0B4D454E5544454641-              db      11,     "MENUDEFAULT",  CONFIG_MENUDEFAULT
45971 00004D6E 554C5441
45972 00004D72 084D454E554954454D-              db      8,      "MENUITEM",     CONFIG_MENUITEM
45972 00004D7B 45
45973                                   ;endif
45974 00004D7C 0A4D554C5449545241-              db      10,     "MULTITRACK",   CONFIG_MULTITRACK
45974 00004D85 434B4D
45975                                   ;ifdef MULTI_CONFIG
45976 00004D88 074E554D4C4F434B4E               db      7,      "NUMLOCK",      CONFIG_NUMLOCK
45977                                   ;endif
45978 00004D91 0352454D30                       db      3,      "REM",          CONFIG_REM
45979                                   ;ifdef MULTI_CONFIG
45980 00004D96 0353455456                       db      3,      "SET",          CONFIG_SET
45981                                   ;endif
45982 00004D9B 055348454C4C53                   db      5,      "SHELL",        CONFIG_SHELL
45983                                   ;if    STACKSW
45984 00004DA2 06535441434B534B                 db      6,      "STACKS",       CONFIG_STACKS
45985                                   ;endif
45986 00004DAA 085357495443484553-              db      8,      "SWITCHES",     CONFIG_SWITCHES
45986 00004DB3 31
45987                                       ; 18/03/2025 (BugFix)
45988                                       ;db     0
45989
45990                                       ; 10/09/2023
45991                                   ;aDosdata:  ; PCDOS 7.1 IBMBIO.COM - SYSINIT:5550h
45992                                       ; 13/04/2024 - Retro DOS v5.0
45993 00004DB4 07444F534441544154           db      7,      "DOSDATA",      CONFIG_DOSDATA ; 'T'
45994 00004DBD 00                           db      0
45995
45996                                   ;%endif ; 02/11/2022
45997
45998                                   ; 01/01/2023 - Retro DOS v4.2
45999                                   %if 0
46000
46001                                   comtab:
46002                                       ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
46003                                       ; (SYSINIT:38EDh)
46004                                       db      7,      "BUFFERS",      CONFIG_BUFFERS
46005                                       db      5,      "BREAK",        CONFIG_BREAK
46006                                       db      6,      "DEVICE",       CONFIG_DEVICE
46007                                       db      10,     "DEVICEHIGH",   CONFIG_DEVICEHIGH
46008                                       db      5,      "FILES",        CONFIG_FILES
46009                                       db      4,      "FCBS",         CONFIG_FCBS
46010                                       db      9,      "LASTDRIVE",    CONFIG_LASTDRIVE
46011                                       db      10,     "MULTITRACK",   CONFIG_MULTITRACK
46012                                       db      8,      "DRIVPARM",     CONFIG_DRIVPARM
46013                                       db      6,      "STACKS",       CONFIG_STACKS
46014                                       db      7,      "COUNTRY",      CONFIG_COUNTRY
46015                                       db      5,      "SHELL",        CONFIG_SHELL
46016                                       db      7,      "INSTALL",      CONFIG_INSTALL
46017                                       db      7,      "COMMENT",      CONFIG_COMMENT
46018                                       db      3,      "REM",          CONFIG_REM
46019                                       db      8,      "SWITCHES",     CONFIG_SWITCHES
46020                                       db      3,      "DOS",          CONFIG_DOS
46021                                       db      0
46022
46023                                   %endif
46024
46025                                   ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
46026                                   ; (SYSINIT:530Ch)
46027
46028                                   ; 13/04/2024 - Retro DOS v5.0
46029                                   ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:555Ah)
46030
46031                                   deviceparameters:
46032                                       ; A_DEVICEPARAMETERS <0,dev_3inch720kb,0,80>
46033                                   devp.specialfunc:  ; deviceparameters +
46034 00004DBE 00                           db      0       ; A_DEVICEPARAMETERS.DP_SPECIALFUNCTIONS
46035                                   devp.devtype:
46036 00004DBF 02                           db      2       ; A_DEVICEPARAMETERS.DP_DEVICETYPE
46037                                   devp.devattr:
46038 00004DC0 0000                         dw      0       ; A_DEVICEPARAMETERS.DP_DEVICEATTRIBUTES
46039                                   devp.cylinders:
46040 00004DC2 5000                         dw      80      ; A_DEVICEPARAMETERS.DP_CYLINDERS
46041
46042                                   ; 04/08/2023 - Retro DOS v4.2 IO.SYS (optimization)
46043
46044                                       ;times 286    db 0
46045                                   devp.mediatype:             ; A_DEVICEPARAMETERS.DP_MEDIATYPE
46046 00004DC4 00                           db      0
46047                                   devp.bpb:           ; A_DEVICEPARAMETERS.DP_BPB
46048                                   devp.bps:           ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.BYTESPERSECTOR
46049 00004DC5 0000                         dw      0
46050                                   devp.secperclus:    ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERCLUSTER
46051 00004DC7 00                           db      0
46052 00004DC8 0000                         dw      0       ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.RESERVEDSECTORS
46053 00004DCA 00                           db      0       ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.NUMBEROFFATS
46054 00004DCB 0000                         dw      0       ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.ROOTENTRIES
46055                                   devp.totalsecs:             ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.TOTALSECTORS
46056 00004DCD 0000                         dw      0
46057                                   devp.mediaid:               ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.MEDIADESCRIPTOR
46058 00004DCF 00                           db      0
46059 00004DD0 0000                         dw      0       ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERFAT
46060                                   devp.spt:           ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.SECTORSPERTRACK
46061 00004DD2 0000                         dw      0
46062                                   devp.heads:         ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.HEADS
46063 00004DD4 0000                         dw      0
46064
46065                                       ; 13/04/2024 - Retro DOS v5.0
46066                                       ; (PCDOS 7.1 IBMBIO.COM)
46067 00004DD6 00<rep 44h>                  times   68 db 0 ; PCDOS 7.1 (FAT32 BPB)
46068                                       ;;times 14 db 0 ; MSDOS 6.21
46069                                       ;dw     0       ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.HIDDENSECTORS
46070                                       ;dw     0
```

```
46071                                   ;dw     0       ; A_DEVICEPARAMETERS.DP_BPB+A_BPB.BIGTOTALSECTORS
46072                                   ;dw     0
46073                                   ;times  6 db 0
46074
46075                                 devp.trktblents:
46076 00004E1A 0000                     dw      0       ; A_DEVICEPARAMETERS.DP_TRACKTABLEENTRIES
46077                                 devp.sectbl:            ; A_DEVICEPARAMETERS.DP_SECTORTABLE
46078 00004E1C 00<rep FCh>              times   252 db 0 ; MAX_SECTORS_IN_TRACK * A_SECTORTABLE.size
46079                                                    ; 63*4 bytes
46080
46081                                 ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
46082                                 ; (SYSINIT:5430h)
46083
46084                                 ; 13/04/2024 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
46085                                 ; (PCDOS 7.1 IBMBIO.COM - SYSINIT:56B4h)
46086
46087 00004F18 0200                  hlim:       dw      2
46088 00004F1A 0900                  slim:       dw      9
46089
46090 00004F1C 00                    drive:      db      0
46091
46092                                 switches:
46093 00004F1D 0000                     dw      0
46094
46095                                 ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
46096                                 ; (SYSINIT:5437h)
46097
46098                                 ; the following are the recommended bpbs for the media that
46099                                 ; we know of so far.
46100
46101                                 ; 02/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
46102                                 ; MSDOS 5.0 IO.SYS - SYSINIT:3AA9h
46103
46104                                 ; 27/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
46105                                 ; PCDOS 7.1 IBMBIO.COM - SYSINIT:56BBh
46106
46107                                 ; 48 tpi diskettes
46108
46109 00004F1F 0002                  bpb48t:     dw      512
46110 00004F21 02                       db      2
46111 00004F22 0100                     dw      1
46112 00004F24 02                       db      2
46113 00004F25 7000                     dw      112
46114 00004F27 D002                     dw      2*9*40 ; 720
46115 00004F29 FD                       db      0FDh
46116 00004F2A 0200                     dw      2
46117 00004F2C 0900                     dw      9
46118 00004F2E 0200                     dw      2
46119 00004F30 00000000                 dd      0
46120 00004F34 00000000                     dd      0
46121                                   ; 27/12/2023
46122 00004F38 00<rep 1Ch>              times   28 db 0             ; FAT32 extensions (to BDS)
46123 00004F54 90                       db      90h
46124
46125                                 ; 96tpi diskettes
46126
46127 00004F55 0002                  bpb96t:     dw      512
46128 00004F57 01                       db      1
46129 00004F58 0100                     dw      1
46130 00004F5A 02                       db      2
46131 00004F5B E000                     dw      224
46132 00004F5D 6009                     dw      2*15*80 ; 2400
46133 00004F5F F9                       db      0F9h
46134 00004F60 0700                     dw      7
46135 00004F62 0F00                     dw      15
46136 00004F64 0200                     dw      2
46137 00004F66 00000000                 dd      0
46138 00004F6A 00000000                     dd      0
46139                                   ; 27/12/2023
46140 00004F6E 00<rep 1Ch>              times   28 db 0             ; FAT32 extensions (to BDS)
46141 00004F8A 90                       db      90h
46142
46143                                 ; 3 1/2 inch diskette bpb
46144
46145 00004F8B 0002                  bpb35:      dw      512
46146 00004F8D 02                       db      2
46147 00004F8E 0100                     dw      1
46148 00004F90 02                       db      2
46149 00004F91 7000                     dw      112
46150 00004F93 A005                     dw      2*9*80 ; 1440
46151 00004F95 F9                       db      0F9h
46152 00004F96 0300                     dw      3
46153 00004F98 0900                     dw      9
46154 00004F9A 0200                     dw      2
46155 00004F9C 00000000                 dd      0
46156 00004FA0 00000000                     dd      0
46157                                   ; 27/12/2023
46158 00004FA4 00<rep 1Ch>              times   28 db 0             ; FAT32 extensions (to BDS)
46159 00004FC0 90                       db      90h
46160
46161 00004FC1 0002                  bpb35h:     dw      512
46162 00004FC3 01                       db      1
46163 00004FC4 0100                     dw      1
46164 00004FC6 02                       db      2
46165 00004FC7 E000                     dw      224
46166 00004FC9 400B                     dw      2*18*80 ; 2880
46167 00004FCB F0                       db      0F0h
46168 00004FCC 0900                     dw      9
46169 00004FCE 1200                     dw      18
46170 00004FD0 0200                     dw      2
46171 00004FD2 00000000                 dd      0
46172 00004FD6 00000000                     dd      0
46173                                   ; 27/12/2023
46174 00004FDA 00<rep 1Ch>              times   28 db 0             ; FAT32 extensions (to BDS)
46175 00004FF6 90                       db      90h
46176
46177                                 ; m037 - BEGIN
46178
46179 00004FF7 0002                  bpb288:     dw      512
46180 00004FF9 02                       db      2
46181 00004FFA 0100                     dw      1
46182 00004FFC 02                       db      2
46183 00004FFD F000                     dw      240
46184 00004FFF 8016                     dw      2*36*80 ; 5760
46185 00005001 F0                       db      0F0h
46186 00005002 0900                     dw      9
46187 00005004 2400                     dw      36
46188 00005006 0200                     dw      2
46189 00005008 00000000                 dd      0
46190 0000500C 00000000                     dd      0
46191                                   ; 27/12/2023
46192 00005010 00<rep 1Ch>              times   28 db 0             ; FAT32 extensions (to BDS)
46193 0000502C 90                       db      90h
46194
```

```
                                ; m037 - END
                                ;
                                ; 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,'$'
00005154 74696F6E20746F6F20-
0000515D 6C617276520666F72-
00005166 206D656D6F72790D0A-
0000516F 24
                                badblock:
00005170 0D0A                       db     0Dh,0Ah
00005172 546F6F206D616E7920-        db     'Too many block devices',0Dh,0Ah,'$'
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
                                ;
```

```
46288 000051C2 4F4E                     OnOff:      db      'ON'
46289 000051C4 4F4646                   OnOff2:     db      'OFF'
46290
46291                                       ; 04/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
46292                                       ; (SYSINIT:5673h)
46293                                   ;StartMsg:
46294                                   ;   db      'Starting MS-DOS...',0Dh,0Ah
46295                                   ;   db      0Ah,0
46296
46297                                       ; 17/12/2023 - Retro DOS v5.0 (Modified PCDOS 7.1 IBMBIO.COM)
46298                                       ; (SYSINIT:58F7h)
46299                                   StartMsg:
46300 000051C7 5374617274696E6720-          db      'Starting PC DOS...',0Dh,0Ah
46300 000051D0 504320444F532E2E2E-
46300 000051D9 0D0A
46301 000051DB 0A00                         db      0Ah,0
46302
46303                                   _$PauseMsg:
46304                                       ; 17/12/2023
46305                                       ;db     'Press any key to continue . . .',0Dh,0Ah,'$'
46306                                       ; 04/08/2023 (PCDOS 7.10 - IBMBIO.COM SYSINIT:590Dh)
46307 000051DD 507265737320616E79-          db      'Press any key to continue...',0Dh,0Ah,'$'
46307 000051E6 206B657920746F2063-
46307 000051EF 6F6E74696E75652E2E-
46307 000051F8 2E0D0A24
46308                                   _$CleanMsg:
46309                                       ;db     'MS-DOS is bypassing your CONFIG.SYS and AUTOEXEC.BAT files.',0Dh,0Ah,'$'
46310                                       ; 17/12/2023
46311 000051FC 504320444F53206973-          db      'PC DOS is bypassing your CONFIG.SYS and AUTOEXEC.BAT files.',0Dh,0Ah,'$'
46311 00005205 20627970617373696E-
46311 0000520E 6720796F757220434F-
46311 00005217 4E4649472E53595320-
46311 00005220 616E64204155544F45-
46311 00005229 5845432E4241542066-
46311 00005232 696C65732E0D0A24
46312                                   _$InterMsg:
46313                                       ;db     'MS-DOS will prompt you to confirm each CONFIG.SYS command.',0Dh,0Ah,'$'
46314                                       ; 17/12/2023
46315 0000523A 504320444F53207769-          db      'PC DOS will prompt you to confirm each CONFIG.SYS command.',0Dh,0Ah,'$'
46315 00005243 6C6C2070726F6D7074-
46315 0000524C 20796F7520746F2063-
46315 00005255 6F6E6669726D206561-
46315 0000525E 636820434F4E464947-
46315 00005267 2E53595320636F6D6D-
46315 00005270 616E642E0D0A24
46316                                   _$MenuHeader:
46317 00005277 0D0A                         db      0Dh,0Ah
46318                                       ; 17/12/2023
46319                                       ;db     '  MS-DOS 6.2 Startup Menu',0Dh,0Ah
46320                                       ;db     '  '
46321                                       ;times  23 db (0CDh)  ; ALT 205 ; '=======================' ; 06/08/2023
46322                                       ;db     0Dh,0Ah,'$'
46323                                       ; 04/08/2023 (PCDOS 7.10 - IBMBIO.COM SYSINIT:59A7h)
46324 00005279 2020504320444F5320-          db      '  PC DOS 7.1 Startup Menu',0Dh,0Ah
46324 00005282 372E312053746172742-
46324 0000528B 7570204D656E750D0A
46325 00005294 2020                         db      '  '
46326 00005296 CD<rep 17h>                  times   23 db (0CDh)  ; ALT 205 ; '=======================' ; 06/08/2023
46327 000052AD 0D0A24                       db      0Dh,0Ah,'$'
46328                                   _$MenuPrmpt:
46329 000052B0 2020456E7465722061-          db      '  Enter a choice: $'
46329 000052B9 2063686F6963653A20-
46329 000052C2 24
46330                                   _$StatusLine:
46331 000052C3 46353D427970617373-          db      'F5=Bypass startup files F8=Confirm each line of CONFIG.SYS '
46331 000052CC 207374617274757020-
46331 000052D5 66696C65732046383D-
46331 000052DE 436F6E6669726D2065-
46331 000052E7 616368206C696E6520-
46331 000052F0 6F6620434F4E464947-
46331 000052F9 2E53595320
46332 000052FE 616E64204155544F45-          db      'and AUTOEXEC.BAT [ ]$'
46332 00005307 5845432E424154205B-
46332 00005310 205D24
46333                                   _$InterPrmpt:
46334                                       ;db     ' [Y,N]?$'
46335                                       ; 13/04/2024
46336                                       ; 04/08/2023
46337 00005313 205B592C4E2C455343-          db      ' [Y,N,ESC]?$' ; PCDOS 7.1 - IBMBIO.COM
46337 0000531C 5D3F24
46338 0000531F 59455324                 _$YES:      db      'YES$'
46339 00005323 4E4F2024                 _$NO:       db      'NO $'
46340                                   _$TimeOut:
46341 00005327 54696D652072656D61-          db      'Time remaining: $'
46341 00005330 696E696E673A2024
46342                                   badcomprmpt:
46343                                       ;db     'Enter correct name of Command Interpreter (eg, C:\COMMAND.COM)'
46344                                       ; 13/04/2024 (PCDOS 7.1 IBMBIO.COM)
46345 00005338 456E74657220636F72-          db      'Enter correct name of Command Interpreter (for example, C:\COMMAND.COM)'
46345 00005341 72656374206E616D65-
46345 0000534A 206F6620436F6D6D61-
46345 00005353 6E6420496E74657270-
46345 0000535C 726574657220028666F-
46345 00005365 72206578616D706C65-
46345 0000536E 2C20433A5C434F4D4D-
46345 00005377 414E442E434F4D29
46346 0000537F 0D0A24                       db      0Dh,0Ah,'$'
46347                                   _$AutoPrmpt:
46348 00005382 50726F636573732041-          db      'Process AUTOEXEC.BAT [Y,N]?$'
46348 0000538B 55544F455845432E42-
46348 00005394 4154205B592C4E5D3F-
46348 0000539D 24
46349
46350                                   ;%endif ; 02/11/2022
46351
46352                                   ; 01/01/2023 - Retro DOS v4.2 (Modified MSDOS 6.21 IO.SYS)
46353                                   ; (SYSINIT:5840h)
46354
46355                                   ; 02/11/2022 - Retro DOS v4.0 (Modified MSDOS 5.0 IO.SYS)
46356                                   ; MSDOS 5.0 IO.SYS - SYSINIT:3CE0h
46357
46358                                   TooManyDrivesMsg:
46359 0000539E 5741524E494E472120-          db      'WARNING! Logical drives past Z: exist and will be ignored',0Dh,0Ah,'$'
46359 000053A7 4C6F676963616C2064-
46359 000053B0 726976657320706173-
46359 000053B9 74205A3A2065786973-
46359 000053C2 7420616E642077696C-
46359 000053CB 6C2062652069676E6F-
46359 000053D4 7265640D0A24
46360
46361                                   ; MSDOS 6.21 IO.SYS - SYSINIT:587Ch
46362                                       ;db     'Wrong DBLSPACE.BIN version',0Dh,0Ah,'$'
46363                                       ;db     7 dup(0)
46364
```

```
46365                                   ;times  7 db 0
46366                                   ; 02/11/2022 (MSDOS 5.0 IO.SYS SYSINIT compatibility)
46367                                 ; MSDOS 5.0 IO.SYS - SYSINIT:3D1Ch
46368                                   ; 09/12/2022
46369                                   ;times 4 db 0
46370
46371                                   ; 08/04/2024 - Retro DOS v5.0
46372                                 ; PCDOS 7.1 IBMBIO.COM - SYSINIT:5B0Bh
46373                                 baddblspace:
46374 000053DA 526571756972656420-      db      'Required system component is not installed',0Dh,0Ah,'$',0
46374 000053E3 73797374656D20636F-
46374 000053EC 6D706F6E656E742069-
46374 000053F5 73206E6F7420696E73-
46374 000053FE 74616C6C65640D0A24-
46374 00005407 00
46375                                   ;db     7 dup(0)
46376
46377                                 ;--------------------------------------------------------------------------
46378                                           ; 09/12/2022
46379                                           ;db 0
46380
46381                                 number3div  equ ($-SYSINIT$)
46382                                 number3mod  equ (number3div % 16)
46383
46384                                 %if number3mod>0 & number3mod<16
46385 00005408 00<rep 8h>                       times (16-number3mod) db 0
46386                                 %endif
46387
46388                                 ;--------------------------------------------------------------------------
46389                                 ; 09/12/2022 - MSDOS 5.0 IO.SYS:3D20h ;;; SI_end = 3D20h for MSDOS 5.0 IO.SYS
46390                                 ;--------------------------------------------------------------------------
46391
46392                                 ;MSDOS 6.21 IO.SYS - SYSINIT:5899h
46393
46394                                 ;--------------------------------------------------------------------------
46395                                 ; 20/04/2019 - Retro DOS v4.0
46396
46397                                 ; 09/12/2022
46398                                 ;
46399                                 ;bss_start:
46400                                 ;
46401                                 ;ABSOLUTE bss_start
46402                                 ;
46403                                 ;alignb 16
46404
46405                                 SI_end:  ; SI_end equ $
46406
46407                                 ;--------------------------------------------------------------------------
46408
46409                                 ;sysinitseg ends
46410
46411                                 ; ****************************************************************************
46412
46413                                 ; 04/01/2023 - MSDOS 6.21 SYSINIT:SI_end = SYSINIT:58A0h (IOSYS:9F46h)
46414                                 ; 09/12/2022 - MSDOS 5.0 SYSINIT:SI_end = SYSINIT:3D20h
46415
46416                                 SYSINITSIZE equ SI_end - SYSINIT$
46417                                 DOSLOADSEG  equ SYSINITSEG+((SYSINITSIZE+15)/16)
46418
46419                                 ;--------------------------------------------------------------------------
46420                                 ; End of Retro DOS 5.0 (PCDOS 7.1) IBMBIO.COM src by Erdogan Tan -21/04/2024-
46421                                 ;--------------------------------------------------------------------------
```