TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0

```
; ****************************************************************************
; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0
; ----------------------------------------------------------------------------
; Last Update: 31/12/2017
; ----------------------------------------------------------------------------
; Beginning: 04/01/2016
; ----------------------------------------------------------------------------
; Assembler: NASM version 2.11 (trdos386.s)
; ----------------------------------------------------------------------------
; Turkish Rational DOS
; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
;
; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
; unix386.s (03/01/2016)
;
; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
; TRDOS2.ASM (09/11/2011)
;
; Derived from 'IBM PC-XT-286' BIOS source code (1986)
; ****************************************************************************
; nasm trdos386.s -l trdos386.txt -o TRDOS386.SYS


KLOAD equ 10000h ; Kernel loading address
      ; NOTE: Retro UNIX 8086 v1 /boot code loads kernel at 1000h:0000h
KCODE equ 08h      ; Code segment descriptor (ring 0)
KDATA equ 10h      ; Data segment descriptor (ring 0)
; 19/03/2015
UCODE equ 1Bh ; 18h + 3h  (ring 3)
UDATA equ 23h ; 20h + 3h  (ring 3)
; 24/03/2015
TSS   equ 28h      ; Task state segment descriptor (ring 0)
; 19/03/2015
CORE  equ 400000h  ; Start of USER's virtual/linear address space
                 ; (at the end of the 1st 4MB)
ECORE equ 0FFC00000h ; End of USER's virtual address space (4GB - 4MB)
                 ; ULIMIT = (ECORE/4096) - 1 = 0FFBFFh (in GDT)

;; 27/12/2013
;KEND    equ KLOAD + 65536 ; (28/12/2013) (end of kernel space)
; 04/07/2016
KEND    equ KERNELFSIZE + KLOAD


; IBM PC/AT BIOS ----- 10/06/85 (postequ.inc)
;--------- CMOS TABLE LOCATION ADDRESS'S -------------------------------------
CMOS_SECONDS EQU    00H          ; SECONDS (BCD)
CMOS_SEC_ALARM     EQU    01H          ; SECONDS ALARM (BCD)
CMOS_MINUTES EQU    02H          ; MINUTES (BCD)
CMOS_MIN_ALARM     EQU    03H          ; MINUTES ALARM (BCD)
CMOS_HOURS   EQU    04H          ; HOURS (BCD
CMOS_HR_ALARM      EQU    005H         ; HOURS ALARM   (BCD)
CMOS_DAY_WEEK      EQU    06H          ; DAY OF THE WEEK  (BCD)
CMOS_DAY_MONTH     EQU    07H          ; DAY OF THE MONTH (BCD)
CMOS_MONTH   EQU    08H          ; MONTH (BCD)
CMOS_YEAR    EQU    09H          ; YEAR (TWO DIGITS) (BCD)
CMOS_CENTURY EQU    32H          ; DATE CENTURY BYTE (BCD)
CMOS_REG_A   EQU    0AH          ; STATUS REGISTER A
CMOS_REG_B   EQU    00BH         ; STATUS REGISTER B  ALARM
CMOS_REG_C   EQU    00CH         ; STATUS REGISTER C  FLAGS
CMOS_REG_D   EQU    0DH          ; STATUS REGISTER D  BATTERY
CMOS_SHUT_DOWN     EQU    0FH          ; SHUTDOWN STATUS COMMAND BYTE
;----------------------------------------
;     CMOS EQUATES FOR THIS SYSTEM      ;
;-----------------------------------------------------------------------------
CMOS_PORT    EQU    070H         ; I/O ADDRESS OF CMOS ADDRESS PORT
CMOS_DATA    EQU    071H         ; I/O ADDRESS OF CMOS DATA PORT
NMI          EQU    10000000B    ; DISABLE NMI INTERRUPTS MASK -
                                 ; HIGH BIT OF CMOS LOCATION ADDRESS

; Memory Allocation Table Address
; 05/11/2014
; 31/10/2014
MEM_ALLOC_TBL      equ    100000h            ; Memory Allocation Table at the end of
                                 ; the 1st 1 MB memory space.
                                 ; (This address must be aligned
                                 ;  on 128 KB boundary, if it will be
                                 ;  changed later.)
                                 ; ((lower 17 bits of 32 bit M.A.T.
                                 ;   address must be ZERO)).
                                 ; ((((Reason: 32 bit allocation
                                 ;     instructions, dword steps)))
                                 ; (((byte >> 12 --> page >> 5)))
;04/11/2014
PDE_A_PRESENT      equ    1            ; Present flag for PDE
PDE_A_WRITE  equ    2            ; Writable (write permission) flag
PDE_A_USER   equ    4            ; User (non-system/kernel) page flag
;
PTE_A_PRESENT      equ    1            ; Present flag for PTE (bit 0)
PTE_A_WRITE  equ    2            ; Writable (write permission) flag (bit 1)
PTE_A_USER   equ    4            ; User (non-system/kernel) page flag (bit 2)
PTE_A_ACCESS     equ       32          ; Accessed flag (bit 5) ; 09/03/2015

; 17/02/2015 (unix386.s)
; 10/12/2014 - 30/12/2014 (0B000h -> 9000h) (dsectrm2.s)
DPT_SEGM equ 09000h  ; FDPT segment (EDD v1.1, EDD v3)
;
HD0_DPT       equ 0     ; Disk parameter table address for hd0
HD1_DPT       equ 32          ; Disk parameter table address for hd1
HD2_DPT       equ 64          ; Disk parameter table address for hd2
HD3_DPT       equ 96          ; Disk parameter table address for hd3

```

```

                                 ; FDPT (Phoenix, Enhanced Disk Drive Specification v1.1, v3.0)
                                 ;      (HDPT: Programmer's Guide to the AMIBIOS, 1993)
                                 ;
                                 FDPT_CYLS    equ 0 ; 1 word, number of cylinders
                                 FDPT_HDS     equ 2 ; 1 byte, number of heads
                                 FDPT_TT            equ 3 ; 1 byte, A0h = translated FDPT with logical values
                                                   ; otherwise it is standard FDPT with physical values
                                 FDPT_PCMP    equ 5 ; 1 word, starting write precompensation cylinder
                                                   ; (obsolete for IDE/ATA drives)
                                 FDPT_CB            equ 8 ; 1 byte, drive control byte
                                                    ; Bits 7-6 : Enable or disable retries (00h = enable)
                                                    ; Bit 5      : 1 = Defect map is located at last cyl. + 1
                                                    ; Bit 4 : Reserved. Always 0
                                                    ; Bit 3 : Set to 1 if more than 8 heads
                                                    ; Bit 2-0 : Reserved. Alsways 0
                                 FDPT_LZ            equ 12 ; 1 word, landing zone (obsolete for IDE/ATA drives)
                                 FDPT_SPT     equ 14 ; 1 byte, sectors per track

                                 ; Floppy Drive Parameters Table (Programmer's Guide to the AMIBIOS, 1993)
                                 ; (11 bytes long) will be used by diskette handler/bios
                                 ; which is derived from IBM PC-AT BIOS (DISKETTE.ASM, 21/04/1986).

                                 ; 01/02/2016
                                 Logical_DOSDisks equ 90000h + 100h ; 26*256 = 6656 bytes
                                 Directory_Buffer equ 80000h ; max = 64K Bytes
                                 FAT_Buffer    equ 91C00h ; 1536 bytes (3 sectors)
                                 ; 15/02/2016
                                 Cluster_Buffer      equ 70000h ; max = 64K Bytes ; buffer for file read & write
                                 ; 11/04/2016
                                 Env_Page:     equ 93000h ; 512 bytes (4096 bytes)
                                 Env_Page_Size       equ 512    ; (4096 bytes)
                                 ; 30/07/2016
                                 Video_Pg_Backup     equ 98000h ; Mode 3h, video page backup (32K, 8 pages)


                                 [BITS 16]       ; We need 16-bit intructions for Real mode

                                 [ORG 0]
                                       ; 12/11/2014
                                       ; Save boot drive number (that is default root drive)
00000000 8816[F25C]                    mov    [boot_drv], dl ; physical drv number

                                       ; Determine installed memory
                                       ; 31/10/2014
                                       ;
00000004 B801E8                        mov    ax, 0E801h ; Get memory size
00000007 CD15                          int    15h       ; for large configurations
00000009 7308                          jnc    short chk_ms
0000000B B488                          mov    ah, 88h    ; Get extended memory size
0000000D CD15                          int    15h
                                       ;
                                       ;mov   al, 17h      ; Extended memory (1K blocks) low byte
                                       ;out   70h, al ; select CMOS register
                                       ;in    al, 71h ; read data (1 byte)
                                       ;mov   cl, al
                                       ;mov   al, 18h ; Extended memory (1K blocks) high byte
                                       ;out   70h, al ; select CMOS register
                                       ;in    al, 71h ; read data (1 byte)
                                       ;mov   ch, al
                                       ;
0000000F 89C1                          mov    cx, ax
00000011 31D2                          xor    dx, dx
                                 chk_ms:
00000013 890E[EE5C]                    mov    [mem_1m_1k], cx
00000017 8916[F05C]                    mov    [mem_16m_64k], dx
                                       ; 05/11/2014
                                       ;and   dx, dx
                                       ;jz    short L2
0000001B 81F90004                        cmp     cx, 1024
0000001F 7351                          jnb    short L0
                                               ; insufficient memory_error
                                               ; Minimum 2 MB memory is needed...
                                       ; 05/11/2014
                                       ; (real mode error printing)
00000021 FB                            sti
00000022 BE[3600]                      mov    si, msg_out_of_memory
00000025 BB0700                        mov    bx, 7
00000028 B40E                          mov    ah, 0Eh      ; write tty
                                 oom_1:
0000002A AC                            lodsb
0000002B 08C0                          or     al, al
0000002D 7404                          jz     short oom_2
0000002F CD10                          int    10h
00000031 EBF7                          jmp    short oom_1
                                 oom_2:
00000033 F4                              hlt
00000034 EBFD                          jmp    short oom_2

                                 ; 20/02/2017
                                 ; 05/11/2014
                                 msg_out_of_memory:
00000036 070D0A                        db     07h, 0Dh, 0Ah
00000039 496E73756666696369-             db      'Insufficient memory !'
00000042 656E74206D656D6F72-
0000004B 792021
0000004E 0D0A                          db     0Dh, 0Ah
                                 _int13h_48h_buffer: ; 07/07/2016
00000050 284D696E696D756D20-           db     '(Minimum 2MB memory is needed.)'
00000059 324D42206D656D6F72-
00000062 79206973206E656564-
0000006B 65642E29
0000006F 0D0A00                        db     0Dh, 0Ah, 0
```

```
                                       ;

                                 L0:
                                 %include 'diskinit.s' ; 07/03/2015
                             <1> ; ****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - diskinit.s
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 09/07/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 24/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Turkish Rational DOS
                             <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
                             <1> ;
                             <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                             <1> ; diskinit.inc (10/07/2015)
                             <1> ;
                             <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)
                             <1> ; ****************************************************************************
                             <1>
                             <1> ; Retro UNIX 386 v1 Kernel - DISKINIT.INC
                             <1> ; Last Modification: 10/07/2015
                             <1>
                             <1> ; DISK I/O SYSTEM INITIALIZATION - Erdogan Tan (Retro UNIX 386 v1 project)
                             <1>
                             <1> ; ///////// DISK I/O SYSTEM STRUCTURE INITIALIZATION ///////////////
                             <1>
                             <1>       ; 10/12/2014 - 02/02/2015 - dsectrm2.s
                             <1> ;L0:
                             <1>       ; 12/11/2014 (Retro UNIX 386 v1 - beginning)
                             <1>       ; Detecting disk drives... (by help of ROM-BIOS)
00000072 BA7F00              <1>       mov    dx, 7Fh
                             <1> L1:
00000075 FEC2                <1>       inc    dl
00000077 B441                <1>       mov    ah, 41h ; Check extensions present
                             <1>                   ; Phoenix EDD v1.1 - EDD v3
00000079 BBAA55              <1>       mov    bx, 55AAh
0000007C CD13                <1>       int    13h
0000007E 721A                <1>       jc     short L2
                             <1>
00000080 81FB55AA            <1>       cmp    bx, 0AA55h
00000084 7514                <1>       jne    short L2
00000086 FE06[F55C]          <1>       inc    byte [hdc]   ; count of hard disks (EDD present)
0000008A 8816[F45C]          <1>         mov     [last_drv], dl  ; last hard disk number
0000008E BB[785C]            <1>       mov    bx, hd0_type - 80h
00000091 01D3                <1>       add    bx, dx
00000093 880F                <1>       mov    [bx], cl ; Interface support bit map in CX
                             <1>                    ; Bit 0 - 1, Fixed disk access subset ready
                             <1>                    ; Bit 1 - 1, Drv locking and ejecting ready
                             <1>                    ; Bit 2 - 1, Enhanced Disk Drive Support
                             <1>                         ;            (EDD) ready (DPTE ready)
                             <1>                    ; Bit 3 - 1, 64bit extensions are present
                             <1>                         ;            (EDD-3)
                             <1>                    ; Bit 4 to 15 - 0, Reserved
00000095 80FA83              <1>       cmp    dl, 83h       ; drive number < 83h
00000098 72DB                <1>       jb     short L1
                             <1> L2:
                             <1>       ; 23/11/2014
                             <1>       ; 19/11/2014
0000009A 30D2                <1>       xor    dl, dl  ; 0
                             <1>       ; 04/02/2016 (esi -> si)
0000009C BE[F65C]            <1>       mov    si, fd0_type
                             <1> L3:
                             <1>       ; 14/01/2015
0000009F 8816[F35C]          <1>       mov    [drv], dl
                             <1>       ;
000000A3 B408                <1>       mov    ah, 08h ; Return drive parameters
000000A5 CD13                <1>       int    13h
000000A7 7210                <1>       jc     short L4
                             <1>             ; BL = drive type (for floppy drives)
                             <1>             ; DL = number of floppy drives
                             <1>             ;
                             <1>             ; ES:DI = Address of DPT from BIOS
                             <1>             ;
000000A9 881C                <1>       mov    [si], bl ;  Drive type
                             <1>                   ; 4 = 1.44 MB, 80 track, 3 1/2"
                             <1>       ; 14/01/2015
000000AB E8BC01              <1>       call   set_disk_parms
                             <1>       ; 10/12/2014
000000AE 81FE[F65C]          <1>       cmp    si, fd0_type
000000B2 7705                <1>       ja     short L4
000000B4 46                  <1>       inc    si ; fd1_type
000000B5 B201                <1>       mov    dl, 1
000000B7 EBE6                <1>       jmp    short L3
                             <1> L4:
                             <1>       ; Older BIOS (INT 13h, AH = 48h is not available)
000000B9 B27F                <1>       mov    dl, 7Fh
                             <1>       ; 24/12/2014 (Temporary)
000000BB 803E[F55C]00        <1>       cmp    byte [hdc], 0 ; EDD present or not ?
000000C0 0F879000            <1>         ja      L10       ; yes, all fixed disk operations
                             <1>                     ; will be performed according to
                             <1>                     ; present EDD specification
                             <1> L6:
000000C4 FEC2                <1>       inc    dl
000000C6 8816[F35C]          <1>         mov     [drv], dl
000000CA 8816[F45C]          <1>         mov     [last_drv], dl ; 14/01/2015
000000CE B408                <1>       mov    ah, 08h ; Return drive parameters
000000D0 CD13                <1>       int    13h    ; (conventional function)
000000D2 0F828601            <1>         jc      L13      ; fixed disk drive not ready
000000D6 8816[F55C]          <1>         mov     [hdc], dl ; number of drives
                             <1>       ;; 14/01/2013
```

```
                              <1>         ;;push cx
000000DA E88D01               <1>         call   set_disk_parms
                              <1>         ;;pop  cx
                              <1>         ;
                              <1>         ;;and  cl, 3Fh        ; sectors per track (bits 0-6)
000000DD 8A16[F35C]           <1>           mov     dl, [drv]
000000E1 BB0401               <1>         mov    bx, 65*4 ; hd0 parameters table (INT 41h)
000000E4 80FA80               <1>         cmp    dl, 80h
000000E7 7603                 <1>         jna    short L7
000000E9 83C314               <1>         add    bx, 5*4        ; hd1 parameters table (INT 46h)
                              <1> L7:
000000EC 31C0                 <1>         xor    ax, ax
000000EE 8ED8                 <1>         mov    ds, ax
000000F0 8B37                 <1>           mov     si, [bx]
000000F2 8B4702               <1>           mov     ax, [bx+2]
000000F5 8ED8                 <1>         mov    ds, ax
000000F7 3A4C0E               <1>           cmp     cl, [si+FDPT_SPT] ; sectors per track
000000FA 0F855A01             <1>           jne     L12 ; invalid FDPT
000000FE BF0000               <1>         mov    di, HD0_DPT
00000101 80FA80               <1>         cmp    dl, 80h
00000104 7603                 <1>         jna    short L8
00000106 BF2000               <1>         mov    di, HD1_DPT
                              <1> L8:
                              <1>         ; 30/12/2014
00000109 B80090               <1>         mov    ax, DPT_SEGM
0000010C 8EC0                 <1>         mov    es, ax
                              <1>         ; 24/12/2014
0000010E B90800               <1>         mov    cx, 8
00000111 F3A5                 <1>         rep    movsw  ; copy 16 bytes to the kernel's DPT location
00000113 8CC8                 <1>         mov    ax, cs
00000115 8ED8                 <1>         mov    ds, ax
                              <1>         ; 02/02/2015
00000117 8A0E[F35C]           <1>           mov     cl, [drv]
0000011B 88CB                 <1>         mov    bl, cl
0000011D B8F001               <1>         mov    ax, 1F0h
00000120 80E301               <1>         and    bl, 1
00000123 7406                 <1>         jz     short L9
00000125 C0E304               <1>         shl    bl, 4
00000128 2D8000               <1>         sub    ax, 1F0h-170h
                              <1> L9:
0000012B AB                   <1>         stosw  ; I/O PORT Base Address (1F0h, 170h)
0000012C 050602               <1>         add    ax, 206h
0000012F AB                   <1>         stosw  ; CONTROL PORT Address (3F6h, 376h)
00000130 88D8                 <1>         mov    al, bl
00000132 04A0                 <1>         add    al, 0A0h
00000134 AA                   <1>         stosb  ; Device/Head Register upper nibble
                              <1>         ;
00000135 FE06[F35C]           <1>         inc    byte [drv]
00000139 BB[785C]             <1>         mov    bx, hd0_type - 80h
0000013C 01CB                 <1>         add    bx, cx
0000013E 800F80               <1>           or      byte [bx], 80h  ; present sign (when lower nibble is 0)
00000141 A0[F55C]             <1>         mov    al, [hdc]
00000144 FEC8                 <1>         dec    al
00000146 0F841201             <1>           jz      L13
0000014A 80FA80               <1>         cmp    dl, 80h
0000014D 0F8673FF             <1>           jna     L6
00000151 E90801               <1>           jmp     L13
                              <1> L10:
00000154 FEC2                 <1>         inc    dl
                              <1>         ; 25/12/2014
00000156 8816[F35C]           <1>         mov    [drv], dl
0000015A B408                 <1>         mov    ah, 08h ; Return drive parameters
0000015C CD13                 <1>         int    13h    ; (conventional function)
0000015E 0F82FA00             <1>           jc      L13
                              <1>         ; 14/01/2015
00000162 8A16[F35C]           <1>         mov    dl, [drv]
00000166 52                   <1>         push   dx
00000167 51                   <1>         push   cx
00000168 E8FF00               <1>         call   set_disk_parms
0000016B 59                   <1>         pop    cx
0000016C 5A                   <1>         pop    dx
                              <1>         ; 06/07/2016 (BugFix for >64K kernel files)
                              <1>         ; 04/02/2016 (esi -> si)
                              <1>         ;mov   si, _end ; 30 byte temporary buffer address
                              <1>         ;              ; at the '_end' of kernel.
                              <1>         ;mov   word [si], 30
                              <1>         ; 06/07/2016
0000016D BE[5000]             <1>         mov    si, _int13h_48h_buffer
                              <1>         ; 09/07/2016
00000170 B81E00               <1>         mov    ax, 001Eh
00000173 8824                 <1>         mov    [si], ah ; 0
00000175 46                   <1>         inc    si
00000176 8904                 <1>         mov    word [si], ax
                              <1>         ; word [si] = 30
                              <1>         ;
00000178 B448                 <1>         mov    ah, 48h        ; Get drive parameters (EDD function)
0000017A CD13                 <1>         int    13h
0000017C 0F82DC00             <1>           jc      L13
                              <1>         ; 04/02/2016 (ebx -> bx)
                              <1>         ; 14/01/2015
00000180 28FF                 <1>         sub    bh, bh
00000182 88D3                 <1>         mov    bl, dl
00000184 80EB80               <1>         sub    bl, 80h
00000187 81C3[F85C]           <1>         add    bx, hd0_type
0000018B 8A07                 <1>         mov    al, [bx]
0000018D 0C80                 <1>         or     al, 80h
0000018F 8807                 <1>         mov    [bx], al
00000191 81EB[F65C]           <1>         sub    bx, hd0_type - 2 ; 15/01/2015
00000195 81C3[425D]           <1>         add    bx, drv.status
00000199 8807                 <1>         mov    [bx], al
                              <1>         ; 04/02/2016 (eax -> ax)
0000019B 8B4410               <1>         mov    ax, [si+16]
0000019E 854412               <1>         test   ax, [si+18]
```

```
000001A1 7412                 <1>       jz     short L10_A0h
                              <1>                    ; 'CHS only' disks on EDD system
                              <1>                    ;  are reported with ZERO disk size
000001A3 81EB[425D]           <1>       sub    bx, drv.status
000001A7 C1E302               <1>       shl    bx, 2
000001AA 81C3[265D]           <1>       add    bx, drv.size ; disk size (in sectors)
000001AE 8907                 <1>       mov    [bx], ax
000001B0 8B4412               <1>       mov    ax, [si+18]
000001B3 8907                 <1>       mov    [bx], ax
                              <1>
                              <1> L10_A0h: ; Jump here to fix a ZERO (LBA) disk size problem
                              <1>        ; for CHS disks (28/02/2015)
                              <1>       ; 30/12/2014
000001B5 BF0000               <1>       mov    di, HD0_DPT
000001B8 88D0                 <1>       mov    al, dl
000001BA 83E003               <1>       and    ax, 3
000001BD C0E005               <1>       shl    al, 5 ; *32
000001C0 01C7                 <1>       add    di, ax
000001C2 B80090               <1>       mov    ax, DPT_SEGM
000001C5 8EC0                 <1>       mov    es, ax
                              <1>       ;
000001C7 88E8                 <1>       mov    al, ch ; max. cylinder number (bits 0-7)
000001C9 88CC                 <1>       mov    ah, cl
000001CB C0EC06               <1>       shr    ah, 6  ; max. cylinder number (bits 8-9)
000001CE 40                   <1>       inc    ax     ; logical cylinders (limit 1024)
000001CF AB                   <1>       stosw
000001D0 88F0                 <1>       mov    al, dh ; max. head number
000001D2 FEC0                 <1>       inc    al
000001D4 AA                   <1>       stosb         ; logical heads (limits 256)
000001D5 B0A0                 <1>       mov    al, 0A0h ; Indicates translated table
000001D7 AA                   <1>       stosb
000001D8 8A440C               <1>       mov    al, [si+12]
000001DB AA                   <1>       stosb          ; physical sectors per track
000001DC 31C0                 <1>       xor    ax, ax
                              <1>       ;dec   ax      ; 02/01/2015
000001DE AB                   <1>       stosw          ; precompensation (obsolete)
                              <1>       ;xor   al, al  ; 02/01/2015
000001DF AA                   <1>       stosb          ; reserved
000001E0 B008                 <1>       mov    al, 8   ; drive control byte
                              <1>                       ; (do not disable retries,
                              <1>                     ; more than 8 heads)
000001E2 AA                   <1>       stosb
000001E3 8B4404               <1>       mov    ax, [si+4]
000001E6 AB                   <1>       stosw          ; physical number of cylinders
                              <1>       ;push  ax      ; 02/01/2015
000001E7 8A4408               <1>       mov    al, [si+8]
000001EA AA                   <1>       stosb          ; physical num. of heads (limit 16)
000001EB 29C0                 <1>       sub    ax, ax
                              <1>       ;pop   ax      ; 02/01/2015
000001ED AB                   <1>       stosw          ; landing zone (obsolete)
000001EE 88C8                 <1>       mov    al, cl  ; logical sectors per track (limit 63)
000001F0 243F                 <1>       and    al, 3Fh
000001F2 AA                   <1>       stosb
                              <1>       ;sub   al, al  ; checksum
                              <1>       ;stosb
                              <1>       ;
000001F3 83C61A               <1>       add    si, 26   ; (BIOS) DPTE address pointer
000001F6 AD                   <1>       lodsw
000001F7 50                   <1>       push   ax      ; (BIOS) DPTE offset
000001F8 AD                   <1>       lodsw
000001F9 50                   <1>       push   ax      ; (BIOS) DPTE segment
                              <1>       ;
                              <1>       ; checksum calculation
000001FA 89FE                 <1>       mov    si, di
000001FC 06                   <1>       push   es
000001FD 1F                   <1>       pop    ds
                              <1>       ;mov   cx, 16
000001FE B90F00               <1>       mov    cx, 15
00000201 29CE                 <1>       sub    si, cx
00000203 30E4                 <1>       xor    ah, ah
                              <1>       ;del   cl
                              <1> L11:
00000205 AC                   <1>       lodsb
00000206 00C4                 <1>       add    ah, al
00000208 E2FB                 <1>       loop   L11
                              <1>       ;
0000020A 88E0                 <1>       mov    al, ah
0000020C F6D8                 <1>       neg    al     ; -x+x = 0
0000020E AA                   <1>       stosb         ; put checksum in byte 15 of the tbl
                              <1>       ;
0000020F 1F                   <1>       pop    ds     ; (BIOS) DPTE segment
00000210 5E                   <1>       pop    si     ; (BIOS) DPTE offset
                              <1>       ;
                              <1>       ; 23/02/2015
00000211 57                   <1>       push   di
                              <1>       ; ES:DI points to DPTE (FDPTE) location
                              <1>       ;mov   cx, 8
00000212 B108                 <1>       mov    cl, 8
00000214 F3A5                 <1>       rep    movsw
                              <1>       ;
                              <1>       ; 23/02/2015
                              <1>       ; (P)ATA drive and LBA validation
                              <1>       ; (invalidating SATA drives and setting
                              <1>       ; CHS type I/O for old type fixed disks)
00000216 5B                   <1>       pop    bx
00000217 8CC8                 <1>       mov    ax, cs
00000219 8ED8                 <1>       mov    ds, ax
0000021B 268B07               <1>       mov    ax, [es:bx]
0000021E 3DF001               <1>       cmp    ax, 1F0h
00000221 7418                 <1>       je     short L11a
00000223 3D7001               <1>       cmp    ax, 170h
00000226 7413                 <1>       je     short L11a
                              <1>       ; invalidation
```

```
                             <1>       ; (because base port address is not 1F0h or 170h)
00000228 30FF                <1>       xor    bh, bh
0000022A 88D3                <1>       mov    bl, dl
0000022C 80EB80              <1>       sub    bl, 80h
0000022F C687[F85C]00        <1>       mov    byte [bx+hd0_type], 0 ; not a valid disk drive !
00000234 808F[445D]F0        <1>         or      byte [bx+drv.status+2], 0F0h ; (failure sign)
00000239 EB14                <1>       jmp    short L11b
                             <1> L11a:
                             <1>       ; LBA validation
0000023B 268A4704            <1>       mov    al, [es:bx+4] ; Head register upper nibble
0000023F A840                <1>       test   al, 40h ; LBA bit (bit 6)
00000241 750C                <1>       jnz    short L11b ; LBA type I/O is OK! (E0h or F0h)
                             <1>       ; force CHS type I/O for this drive (A0h or B0h)
00000243 28FF                <1>       sub    bh, bh
00000245 88D3                <1>       mov    bl, dl
00000247 80EB80              <1>       sub    bl, 80h ; 26/02/2015
0000024A 80A7[445D]FE        <1>         and     byte [bx+drv.status+2], 0FEh ; clear bit 0
                             <1>                           ; bit 0 = LBA ready bit
                             <1>       ; 'diskio' procedure will check this bit !
                             <1> L11b:
0000024F 3A16[F45C]          <1>       cmp    dl, [last_drv] ; 25/12/2014
00000253 7307                <1>         jnb     short L13
00000255 E9FCFE              <1>         jmp     L10
                             <1> L12:
                             <1>       ; Restore data registers
00000258 8CC8                <1>       mov    ax, cs
0000025A 8ED8                <1>       mov    ds, ax
                             <1> L13:
                             <1>       ; 13/12/2014
0000025C 0E                  <1>       push   cs
0000025D 07                  <1>       pop    es
                             <1> L14:
0000025E B411                <1>       mov    ah, 11h
00000260 CD16                <1>       int    16h
00000262 7466                <1>       jz     short L16 ; no keys in keyboard buffer
00000264 B010                <1>       mov    al, 10h
00000266 CD16                <1>       int    16h
00000268 EBF4                <1>       jmp    short L14
                             <1>
                             <1> set_disk_parms:
                             <1>       ; 04/02/2016 (ebx -> bx)
                             <1>       ; 10/07/2015
                             <1>       ; 14/01/2015
                             <1>       ;push  bx
0000026A 28FF                <1>       sub    bh, bh
0000026C 8A1E[F35C]          <1>       mov    bl, [drv]
00000270 80FB80              <1>       cmp    bl, 80h
00000273 7203                <1>       jb     short sdp0
00000275 80EB7E              <1>       sub    bl, 7Eh
                             <1> sdp0:
00000278 81C3[425D]          <1>       add    bx, drv.status
0000027C C60780              <1>       mov    byte [bx], 80h ; 'Present' flag
                             <1>       ;
0000027F 88E8                <1>       mov    al, ch ; last cylinder (bits 0-7)
00000281 88CC                <1>       mov    ah, cl ;
00000283 C0EC06              <1>       shr    ah, 6  ; last cylinder (bits 8-9)
00000286 81EB[425D]          <1>       sub    bx, drv.status
0000028A D0E3                <1>       shl    bl, 1
0000028C 81C3[FC5C]          <1>       add    bx, drv.cylinders
00000290 40                  <1>       inc    ax  ; convert max. cyl number to cyl count
00000291 8907                <1>       mov    [bx], ax
00000293 50                  <1>       push   ax ; ** cylinders
00000294 81EB[FC5C]          <1>       sub    bx, drv.cylinders
00000298 81C3[0A5D]          <1>       add    bx, drv.heads
0000029C 30E4                <1>       xor    ah, ah
0000029E 88F0                <1>       mov    al, dh ; heads
000002A0 40                  <1>       inc    ax
000002A1 8907                <1>       mov    [bx], ax
000002A3 81EB[0A5D]          <1>         sub     bx, drv.heads
000002A7 81C3[185D]          <1>         add     bx, drv.spt
000002AB 30ED                <1>       xor    ch, ch
000002AD 80E13F              <1>       and    cl, 3Fh      ; sectors (bits 0-6)
000002B0 890F                <1>       mov    [bx], cx
000002B2 81EB[185D]          <1>         sub     bx, drv.spt
000002B6 D1E3                <1>       shl    bx, 1
000002B8 81C3[265D]          <1>       add    bx, drv.size ; disk size (in sectors)
                             <1>       ; LBA size = cylinders * heads * secpertrack
000002BC F7E1                <1>       mul    cx
000002BE 89C2                <1>       mov    dx, ax ; heads*spt
000002C0 58                  <1>       pop    ax ; ** cylinders
000002C1 48                  <1>       dec    ax ; 1 cylinder reserved (!?)
000002C2 F7E2                <1>       mul    dx ; cylinders * (heads*spt)
000002C4 8907                <1>       mov    [bx], ax
000002C6 895702              <1>       mov    [bx+2], dx
                             <1>       ;
                             <1>       ;pop   bx
000002C9 C3                  <1>       retn
                             <1>
                             <1> L16:  ; 28/05/2016

                                       ; 10/11/2014
000002CA FA                            cli    ; Disable interrupts (clear interrupt flag)
                                              ; Reset Interrupt MASK Registers (Master&Slave)
                                       ;mov   al, 0FFh     ; mask off all interrupts
                                       ;out   21h, al             ; on master PIC (8259)
                                       ;jmp   $+2  ; (delay)
                                       ;out   0A1h, al     ; on slave PIC (8259)
                                       ;
                                       ; Disable NMI
000002CB B080                          mov    al, 80h
000002CD E670                          out    70h, al             ; set bit 7 to 1 for disabling NMI
                                       ;23/02/2015
                                       ;nop               ;
                                       ;in    al, 71h             ; read in 71h just after writing out to 70h
                                                           ; for preventing unknown state (!?)
                                       ;
                                       ; 20/08/2014
                                       ; Moving the kernel 64 KB back (to physical address 0)
                                       ; DS = CS = 1000h
                                       ; 05/11/2014
000002CF 31C0                          xor    ax, ax
000002D1 8EC0                          mov    es, ax ; ES = 0
                                       ;
                                       ; 04/07/2016 -  TRDOS 386 (64K - 128K kernel)
000002D3 31F6                                xor    si, si
000002D5 31FF                          xor    di, di
000002D7 B90040                        mov    cx, 16384
000002DA F366A5                        rep    movsd
                                       ;
000002DD 06                            push   es ; 0
000002DE 68[E202]                      push   L17
000002E1 CB                            retf
                                 L17:
000002E2 B90010                        mov    cx, 1000h
000002E5 8EC1                          mov    es, cx  ; 1000h
000002E7 01C9                          add    cx, cx
000002E9 8ED9                          mov    ds, cx  ; 2000h
000002EB 29F6                          sub    si, si
000002ED 29FF                          sub    di, di
000002EF B90040                        mov    cx, 16384
000002F2 F366A5                        rep    movsd

                                       ; Turn off the floppy drive motor
000002F5 BAF203                          mov     dx, 3F2h
000002F8 EE                              out     dx, al ; 0 ; 31/12/2013

                                       ; Enable access to memory above one megabyte
                                 L18:
000002F9 E464                          in     al, 64h
000002FB A802                          test   al, 2
000002FD 75FA                            jnz     short L18
000002FF B0D1                          mov    al, 0D1h     ; Write output port
00000301 E664                          out    64h, al
                                 L19:
00000303 E464                          in     al, 64h
00000305 A802                          test   al, 2
00000307 75FA                            jnz     short L19
00000309 B0DF                          mov    al, 0DFh     ; Enable A20 line
0000030B E660                          out    60h, al
                                 ;L20:
                                       ;
                                       ; Load global descriptor table register

                                         ;mov     ax, cs
                                         ;mov     ds, ax

0000030D 2E0F0116[605C]                  lgdt    [cs:gdtd]

00000313 0F20C0                          mov     eax, cr0
                                       ; or   eax, 1
00000316 40                            inc      ax
00000317 0F22C0                        mov      cr0, eax

                                       ; Jump to 32 bit code

0000031A 66                            db 66h                    ; Prefix for 32-bit
0000031B EA                            db 0EAh            ; Opcode for far jump
0000031C [22030000]                    dd StartPM          ; Offset to start, 32-bit
                                                           ; (1000h:StartPM = StartPM + 10000h)
00000320 0800                          dw KCODE           ; This is the selector for CODE32_DESCRIPTOR,
                                                           ; assuming that StartPM resides in code32

                                 ; 20/02/2017


                                 [BITS 32]

                                 StartPM:
                                       ; Kernel Base Address = 0 ; 30/12/2013
00000322 66B81000                      mov ax, KDATA           ; Save data segment identifier
00000326 8ED8                            mov ds, ax              ; Move a valid data segment into DS register
00000328 8EC0                                mov es, ax              ; Move data segment into ES register
0000032A 8EE0                                mov fs, ax              ; Move data segment into FS register
0000032C 8EE8                                mov gs, ax              ; Move data segment into GS register
0000032E 8ED0                            mov ss, ax              ; Move data segment into SS register
00000330 BC00000900                      mov esp, 90000h         ; Move the stack pointer to 090000h

                                 clear_bss: ; Clear uninitialized data area
                                       ; 11/03/2015
00000335 31C0                          xor  eax, eax ; 0
00000337 B9136F0000                    mov  ecx, (bss_end - bss_start)/4
                                       ;shr  ecx, 2 ; bss section is already aligned for double words
0000033C BF[4E550100]                  mov  edi, bss_start
00000341 F3AB                          rep  stosd

                                 memory_init:
                                       ; Initialize memory allocation table and page tables
                                       ; 16/11/2014
                                       ; 15/11/2014
                                       ; 07/11/2014
                                       ; 06/11/2014
                                       ; 05/11/2014
                                       ; 04/11/2014
                                       ; 31/10/2014 (Retro UNIX 386 v1 - Beginning)
                                       ;
                                 ;     xor    eax, eax
                                 ;     xor    ecx, ecx
00000343 B108                          mov    cl, 8
00000345 BF00001000                    mov    edi, MEM_ALLOC_TBL
0000034A F3AB                          rep    stosd           ; clear Memory Allocation Table
                                                              ; for the first 1 MB memory
                                       ;
0000034C 668B0D[EE5C0000]              mov    cx, [mem_1m_1k]       ; Number of contiguous KB between
                                                              ; 1 and 16 MB, max. 3C00h = 15 MB.
00000353 66C1E902                      shr    cx, 2           ; convert 1 KB count to 4 KB count
00000357 890D[40580100]                mov    [free_pages], ecx
0000035D 668B15[F05C0000]              mov    dx, [mem_16m_64k]  ; Number of contiguous 64 KB blocks
                                                              ; between 16 MB and 4 GB.
00000364 6609D2                        or     dx, dx
00000367 7413                          jz     short mi_0
                                       ;
00000369 6689D0                        mov    ax, dx
0000036C C1E004                        shl    eax, 4          ; 64 KB -> 4 KB (page count)
0000036F 0105[40580100]                add    [free_pages], eax
00000375 0500100000                    add    eax, 4096       ; 16 MB = 4096 pages
0000037A EB07                          jmp    short mi_1
                                 mi_0:
0000037C 6689C8                        mov    ax, cx
0000037F 66050001                      add    ax, 256               ; add 256 pages for the first 1 MB
                                 mi_1:
00000383 A3[3C580100]                  mov    [memory_size], eax ; Total available memory in pages
                                                              ; 1 alloc. tbl. bit = 1 memory page
                                                              ; 32 allocation bits = 32 mem. pages
                                       ;
00000388 05FF7F0000                    add    eax, 32767      ; 32768 memory pages per 1 M.A.T. page
0000038D C1E80F                        shr    eax, 15               ; ((32768 * x) + y) pages (y < 32768)
                                                              ;  --> x + 1 M.A.T. pages, if y > 0
                                                              ;  --> x M.A.T. pages, if y = 0
00000390 66A3[50580100]                mov    [mat_size], ax        ; Memory Alloc. Table Size in pages
00000396 C1E00C                        shl    eax, 12               ; 1 M.A.T. page = 4096 bytes
                                       ;                      ; Max. 32 M.A.T. pages (4 GB memory)
00000399 89C3                          mov    ebx, eax        ; M.A.T. size in bytes
                                       ; Set/Calculate Kernel's Page Directory Address
0000039B 81C300001000                  add    ebx, MEM_ALLOC_TBL
000003A1 891D[38580100]                mov    [k_page_dir], ebx  ; Kernel's Page Directory address
                                                              ; just after the last M.A.T. page
                                       ;
000003A7 83E804                        sub    eax, 4          ; convert M.A.T. size to offset value
000003AA A3[48580100]                  mov    [last_page], eax   ; last page ofset in the M.A.T.
                                       ;                      ; (allocation status search must be
                                                              ; stopped after here)
000003AF 31C0                          xor    eax, eax
000003B1 48                            dec    eax             ; FFFFFFFFh (set all bits to 1)
000003B2 6651                          push   cx
000003B4 C1E905                        shr    ecx, 5          ; convert 1 - 16 MB page count to
                                                              ; count of 32 allocation bits
000003B7 F3AB                          rep    stosd
000003B9 6659                          pop    cx
000003BB 40                            inc    eax             ; 0
000003BC 80E11F                        and    cl, 31          ; remain bits
000003BF 7412                          jz     short mi_4
000003C1 8907                          mov    [edi], eax      ; reset
                                 mi_2:
000003C3 0FAB07                        bts    [edi], eax      ; 06/11/2014
000003C6 FEC9                          dec    cl
000003C8 7404                          jz     short mi_3
000003CA FEC0                          inc    al
000003CC EBF5                          jmp    short mi_2
                                 mi_3:
000003CE 28C0                          sub    al, al          ; 0
000003D0 83C704                        add    edi, 4          ; 15/11/2014
                                 mi_4:
000003D3 6609D2                        or     dx, dx         ; check 16M to 4G memory space
000003D6 7421                          jz     short mi_6     ; max. 16 MB memory, no more...
                                       ;
000003D8 B900021000                    mov    ecx, MEM_ALLOC_TBL + 512 ; End of first 16 MB memory
                                       ;
000003DD 29F9                          sub    ecx, edi       ; displacement (to end of 16 MB)
000003DF 7406                          jz     short mi_5     ; jump if EDI points to
                                                             ;         end of first 16 MB
000003E1 D1E9                          shr    ecx, 1         ; convert to dword count
000003E3 D1E9                          shr    ecx, 1         ; (shift 2 bits right)
000003E5 F3AB                          rep    stosd          ; reset all bits for reserved pages
                                                             ; (memory hole under 16 MB)
                                 mi_5:
000003E7 6689D1                        mov    cx, dx         ; count of 64 KB memory blocks
000003EA D1E9                          shr    ecx, 1         ; 1 alloc. dword per 128 KB memory
000003EC 9C                            pushf                 ; 16/11/2014
000003ED 48                            dec    eax            ; FFFFFFFFh (set all bits to 1)
000003EE F3AB                          rep    stosd
000003F0 40                            inc    eax            ; 0
000003F1 9D                            popf                  ; 16/11/2014
000003F2 7305                          jnc    short mi_6
000003F4 6648                          dec    ax             ; eax = 0000FFFFh
000003F6 AB                            stosd
000003F7 6640                          inc    ax             ; 0
                                 mi_6:
000003F9 39DF                          cmp    edi, ebx       ; check if EDI points to
000003FB 730A                          jnb    short mi_7     ; end of memory allocation table
                                       ;                     ; (>= MEM_ALLOC_TBL + 4906)
000003FD 89D9                          mov    ecx, ebx       ; end of memory allocation table
000003FF 29F9                          sub    ecx, edi       ; convert displacement/offset
00000401 D1E9                          shr    ecx, 1         ; to dword count
00000403 D1E9                          shr    ecx, 1         ; (shift 2 bits right)
00000405 F3AB                          rep    stosd          ; reset all remain M.A.T. bits
                                 mi_7:
                                       ; Reset M.A.T. bits in M.A.T. (allocate M.A.T. pages)
00000407 BA00001000                    mov    edx, MEM_ALLOC_TBL
                                       ;sub   ebx, edx       ; Mem. Alloc. Tbl. size in bytes
                                       ;shr   ebx, 12               ; Mem. Alloc. Tbl. size in pages
0000040C 668B0D[50580100]              mov    cx, [mat_size]        ; Mem. Alloc. Tbl. size in pages
00000413 89D7                          mov    edi, edx
00000415 C1EF0F                        shr    edi, 15               ; convert M.A.T. address to
                                                             ; byte offset in M.A.T.
                                                             ; (1 M.A.T. byte points to
                                                             ;          32768 bytes)
                                                             ; Note: MEM_ALLOC_TBL address
                                                             ; must be aligned on 128 KB
                                                             ; boundary!
00000418 01D7                          add    edi, edx       ; points to M.A.T.'s itself
                                       ; eax = 0
0000041A 290D[40580100]                sub    [free_pages], ecx ; 07/11/2014
                                 mi_8:
00000420 0FB307                        btr    [edi], eax     ; clear bit 0 to bit x (1 to 31)
                                       ;dec   bl
00000423 FEC9                          dec    cl
00000425 7404                          jz     short mi_9
00000427 FEC0                          inc    al
00000429 EBF5                          jmp    short mi_8
                                 mi_9:
                                       ;
                                       ; Reset Kernel's Page Dir. and Page Table bits in M.A.T.
                                       ;            (allocate pages for system page tables)

                                       ; edx = MEM_ALLOC_TBL
0000042B 8B0D[3C580100]                mov    ecx, [memory_size] ; memory size in pages (PTEs)
00000431 81C1FF030000                  add    ecx, 1023     ; round up (1024 PTEs per table)
00000437 C1E90A                        shr    ecx, 10              ; convert memory page count to
                                                            ; page table count (PDE count)
                                       ;
0000043A 51                            push   ecx           ; (**) PDE count (<= 1024)
                                       ;
0000043B 41                            inc    ecx           ; +1 for kernel page directory
                                       ;
0000043C 290D[40580100]                sub    [free_pages], ecx ; 07/11/2014
                                       ;
00000442 8B35[38580100]                mov    esi, [k_page_dir] ; Kernel's Page Directory address
00000448 C1EE0C                        shr    esi, 12              ; convert to page number
                                 mi_10:
0000044B 89F0                          mov    eax, esi      ; allocation bit offset
0000044D 89C3                          mov    ebx, eax
0000044F C1EB03                        shr    ebx, 3        ; convert to alloc. byte offset
00000452 80E3FC                        and    bl,  0FCh     ; clear bit 0 and bit 1
                                                            ;   to align on dword boundary
00000455 83E01F                        and    eax, 31              ; set allocation bit position
                                                            ;  (bit 0 to bit 31)
                                       ;
00000458 01D3                          add    ebx, edx      ; offset in M.A.T. + M.A.T. address
                                       ;
0000045A 0FB303                        btr    [ebx], eax    ; reset relevant bit (0 to 31)
                                       ;
0000045D 46                            inc    esi           ; next page table
0000045E E2EB                          loop   mi_10         ; allocate next kernel page table
                                                            ; (ecx = page table count + 1)
                                       ;
00000460 59                            pop    ecx           ; (**) PDE count (= pg. tbl. count)
                                       ;
                                       ; Initialize Kernel Page Directory and Kernel Page Tables
                                       ;
                                       ; Initialize Kernel's Page Directory
00000461 8B3D[38580100]                mov    edi, [k_page_dir]
00000467 89F8                          mov    eax, edi
00000469 0C03                          or     al, PDE_A_PRESENT + PDE_A_WRITE
                                                          ; supervisor + read&write + present
0000046B 89CA                          mov    edx, ecx     ; (**) PDE count (= pg. tbl. count)
                                 mi_11:
0000046D 0500100000                    add    eax, 4096    ; Add page size (PGSZ)
                                                            ; EAX points to next page table
00000472 AB                            stosd
00000473 E2F8                          loop   mi_11
00000475 29C0                          sub    eax, eax     ; Empty PDE
00000477 66B90004                      mov    cx, 1024     ; Entry count (PGSZ/4)
0000047B 29D1                          sub    ecx, edx
0000047D 7402                          jz     short mi_12
0000047F F3AB                          rep    stosd        ; clear remain (empty) PDEs
                                       ;
                                       ; Initialization of Kernel's Page Directory is OK, here.
                                 mi_12:
                                       ; Initialize Kernel's Page Tables
                                       ;
                                       ; (EDI points to address of page table 0)
                                       ; eax = 0
00000481 8B0D[3C580100]                mov    ecx, [memory_size] ; memory size in pages
00000487 89CA                          mov    edx, ecx     ; (***)
00000489 B003                          mov    al, PTE_A_PRESENT + PTE_A_WRITE
                                                         ; supervisor + read&write + present
                                 mi_13:
0000048B AB                            stosd
0000048C 0500100000                    add    eax, 4096
00000491 E2F8                          loop   mi_13
00000493 6681E2FF03                    and    dx, 1023     ; (***)
00000498 740B                          jz     short mi_14
0000049A 66B90004                      mov    cx, 1024
0000049E 6629D1                        sub    cx, dx       ; from dx (<= 1023) to 1024
000004A1 31C0                          xor    eax, eax
000004A3 F3AB                          rep    stosd        ; clear remain (empty) PTEs
                                                           ; of the last page table
                                 mi_14:
                                       ;  Initialization of Kernel's Page Tables is OK, here.
                                       ;
000004A5 89F8                          mov    eax, edi     ; end of the last page table page
                                                            ; (begining of user space pages)
```

```
000004A7 C1E80F                        shr    eax, 15             ; convert to M.A.T. byte offset
000004AA 24FC                          and    al, 0FCh     ; clear bit 0 and bit 1 for
                                                           ; aligning on dword boundary

000004AC A3[4C580100]                  mov    [first_page], eax
000004B1 A3[44580100]                  mov    [next_page], eax ; The first free page pointer
                                                            ; for user programs
                                                            ; (Offset in Mem. Alloc. Tbl.)
                                       ;
                                       ; Linear/FLAT (1 to 1) memory paging for the kernel is OK, here.
                                       ;

                                       ; Enable paging
                                       ;
000004B6 A1[38580100]                    mov     eax, [k_page_dir]
000004BB 0F22D8                        mov    cr3, eax
000004BE 0F20C0                        mov    eax, cr0
000004C1 0D00000080                    or     eax, 80000000h      ; set paging bit (bit 31)
000004C6 0F22C0                        mov    cr0, eax
                                         ;jmp    KCODE:StartPMP

000004C9 EA                            db 0EAh             ; Opcode for far jump
000004CA [D0040000]                      dd StartPMP              ; 32 bit offset
000004CE 0800                          dw KCODE            ; kernel code segment descriptor


                                 StartPMP:
                                       ; 06/11//2014
                                       ; Clear video page 0
                                       ;
                                       ; Temporary Code
                                       ;
000004D0 B9E8030000                    mov    ecx, 80*25/2
000004D5 BF00800B00                    mov    edi, 0B8000h
                                       ; 30/01/2016
                                       ;xor   eax, eax     ; black background, black fore color
000004DA B800070007                    mov    eax, 07000700h  ; black background, light gray fore color
000004DF F3AB                          rep    stosd

                                       ; 19/08/2014
                                       ; Kernel Base Address = 0
                                       ; It is mapped to (physically) 0 in the page table.
                                       ; So, here is exactly 'StartPMP' address.

                                       ; 29/01/2016 (TRDOS 386 = TRDOS v2.0)
000004E1 BE[8D190100]                  mov    esi, starting_msg
                                       ;; 14/08/2015 (kernel version message will appear
                                       ;;            when protected mode and paging is enabled)
000004E6 BF00800B00                    mov    edi, 0B8000h ; 27/08/2014
000004EB B40A                          mov    ah, 0Ah ; Black background, light green forecolor
                                       ; 20/08/2014
000004ED E88F010000                    call   printk

                                       ; 'UNIX v7/x86' source code by Robert Nordier (1999)
                                       ; // Set IRQ offsets
                                       ;
                                       ;  Linux (v0.12) source code by Linus Torvalds (1991)
                                       ;
                                                                  ;; ICW1
000004F2 B011                          mov    al, 11h                   ; Initialization sequence
000004F4 E620                          out    20h, al                   ;      8259A-1
                                       ; jmp  $+2
000004F6 E6A0                          out    0A0h, al            ;     8259A-2
                                                                  ;; ICW2
000004F8 B020                          mov    al, 20h                   ; Start of hardware ints (20h)
000004FA E621                          out    21h, al                   ;      for 8259A-1
                                       ; jmp  $+2
000004FC B028                          mov    al, 28h                   ; Start of hardware ints (28h)
000004FE E6A1                          out    0A1h, al            ;     for 8259A-2
                                                                  ;
00000500 B004                          mov    al, 04h                   ;; ICW3
00000502 E621                          out    21h, al                   ;      IRQ2 of 8259A-1 (master)
                                       ; jmp  $+2
00000504 B002                          mov    al, 02h                   ;      is 8259A-2 (slave)
00000506 E6A1                          out    0A1h, al            ;
                                                                  ;; ICW4
00000508 B001                          mov    al, 01h                   ;
0000050A E621                          out    21h, al                   ;      8086 mode, normal EOI
                                       ; jmp  $+2
0000050C E6A1                          out    0A1h, al            ;     for both chips.

                                       ;mov   al, 0FFh     ; mask off all interrupts for now
                                       ;out   21h, al
                                       ;; jmp       $+2
                                       ;out   0A1h, al

                                       ; 02/04/2015
                                       ; 26/03/2015 System call (INT 30h) modification
                                       ;  DPL = 3 (Interrupt service routine can be called from user mode)
                                       ;
                                       ;; Linux (v0.12) source code by Linus Torvalds (1991)
                                       ;  setup_idt:
                                       ;
                                         ;; 16/02/2015
                                       ;;mov     dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
                                       ; 21/08/2014 (timer_int)
0000050E BE[50160100]                  mov    esi, ilist
00000513 8D3D[50550100]                lea    edi, [idt]
                                       ; 26/03/2015
00000519 B930000000                    mov    ecx, 48             ; 48 hardware interrupts (INT 0 to INT 2Fh)
                                       ; 02/04/2015
0000051E BB00000800                    mov    ebx,  80000h
                                 rp_sidt1:
```

```
00000523 AD                            lodsd
00000524 89C2                          mov    edx, eax
00000526 66BA008E                      mov    dx, 8E00h
0000052A 6689C3                        mov    bx, ax
0000052D 89D8                          mov    eax, ebx     ; /* selector = 0x0008 = cs */
                                                                    ; /* interrupt gate - dpl=0, present */
0000052F AB                            stosd  ; selector & offset bits 0-15
00000530 89D0                          mov    eax, edx
00000532 AB                            stosd  ; attributes & offset bits 16-23
00000533 E2EE                          loop   rp_sidt1
                                       ; 15/04/2016
                                       ; TRDOS 386 (TRDOS v2.0) /// 32 sofware interrupts ///
                                       ;mov   cl, 16        ; 16 software interrupts (INT 30h to INT 3Fh)
00000535 B120                          mov    cl, 32       ; 32 software interrupts (INT 30h to INT 4Fh)
                                 rp_sidt2:
00000537 AD                            lodsd
00000538 21C0                          and    eax, eax
0000053A 7413                          jz     short rp_sidt3
0000053C 89C2                          mov    edx, eax
0000053E 66BA00EE                      mov    dx, 0EE00h   ; P=1b/DPL=11b/01110b
00000542 6689C3                        mov    bx, ax
00000545 89D8                          mov    eax, ebx     ; selector & offset bits 0-15
00000547 AB                            stosd
00000548 89D0                          mov    eax, edx
0000054A AB                            stosd
0000054B E2EA                          loop   rp_sidt2
0000054D EB16                          jmp    short sidt_OK
                                 rp_sidt3:
0000054F B8[AA0A0000]                  mov    eax, ignore_int
00000554 89C2                          mov    edx, eax
00000556 66BA00EE                      mov    dx, 0EE00h   ; P=1b/DPL=11b/01110b
0000055A 6689C3                        mov    bx, ax
0000055D 89D8                          mov    eax, ebx     ; selector & offset bits 0-15
                                 rp_sidt4:
0000055F AB                            stosd
00000560 92                            xchg   eax, edx
00000561 AB                            stosd
00000562 92                            xchg   edx, eax
00000563 E2FA                          loop   rp_sidt4
                                 sidt_OK:
00000565 0F011D[665C0000]              lidt   [idtd]
                                       ;
                                       ; TSS descriptor setup ; 24/03/2015
0000056C B8[D0570100]                  mov    eax, task_state_segment
00000571 66A3[5A5C0000]                mov    [gdt_tss0], ax
00000577 C1C010                        rol    eax, 16
0000057A A2[5C5C0000]                  mov    [gdt_tss1], al
0000057F 8825[5F5C0000]                mov    [gdt_tss2], ah
00000585 66C705[36580100]68-           mov    word [tss.IOPB], tss_end - task_state_segment
0000058D 00
                                              ;
                                              ; IO Map Base address (When this address points
                                              ; to end of the TSS, CPU does not use IO port
                                              ; permission bit map for RING 3 IO permissions,
                                              ; access to any IO ports in ring 3 will be forbidden.)
                                              ;
                                       ;mov   [tss.esp0], esp ; TSS offset 4
                                       ;mov   word [tss.ss0], KDATA ; TSS offset 8 (SS)
0000058E 66B82800                      mov    ax, TSS  ; It is needed when an interrupt
                                                      ; occurs (or a system call -software INT- is requested)
                                                      ; while cpu running in ring 3 (in user mode).
                                                      ; (Kernel stack pointer and segment will be loaded
                                                      ; from offset 4 and 8 of the TSS, by the CPU.)
00000592 0F00D8                        ltr    ax  ; Load task register
                                       ;
                                 esp0_set0:
                                       ; 30/07/2015
00000595 8B0D[3C580100]                mov    ecx, [memory_size] ; memory size in pages
0000059B C1E10C                        shl    ecx, 12 ; convert page count to byte count
0000059E 81F900004000                  cmp    ecx, CORE ; beginning of user's memory space (400000h)
                                                       ; (kernel mode virtual address)
000005A4 7605                          jna    short esp0_set1
                                       ;
                                       ; If available memory > CORE (end of the 1st 4 MB)
                                       ; set stack pointer to CORE
                                       ;(Because, PDE 0 is reserved for kernel space in user's page directory)
                                       ;(PDE 0 points to page table of the 1st 4 MB virtual address space)
000005A6 B900004000                    mov    ecx, CORE
                                 esp0_set1:
000005AB 89CC                          mov    esp, ecx ; top of kernel stack (**tss.esp0**)
                                 esp0_set_ok:
                                       ; 30/07/2015 (**tss.esp0**)
000005AD 8925[D4570100]                mov    [tss.esp0], esp
000005B3 66C705[D8570100]10-             mov     word [tss.ss0], KDATA
000005BB 00
                                       ; 14/08/2015
                                       ; 10/11/2014 (Retro UNIX 386 v1 - Erdogan Tan)
                                       ;
                                       ;cli   ; Disable interrupts (for CPU)
                                       ;    (CPU will not handle hardware interrupts, except NMI!)
                                       ;
000005BC 30C0                          xor    al, al       ; Enable all hardware interrupts!
000005BE E621                          out    21h, al             ; (IBM PC-AT compatibility)
000005C0 EB00                          jmp    $+2          ; (All conventional PC-AT hardware
000005C2 E6A1                          out    0A1h, al     ;  interrupts will be in use.)
                                                           ; (Even if related hardware component
                                                           ;  does not exist!)
                                       ; Enable NMI
000005C4 B07F                          mov    al, 7Fh             ; Clear bit 7 to enable NMI (again)
000005C6 E670                          out    70h, al
                                       ; 23/02/2015
000005C8 90                            nop
000005C9 E471                          in     al, 71h             ; read in 71h just after writing out to 70h
```

```
                                                           ; for preventing unknown state (!?)
                                       ;
                                       ; Only a NMI can occur here... (Before a 'STI' instruction)
                                       ;
                                       ; 02/09/2014
000005CB 6631DB                        xor    bx, bx
000005CE 66BA0002                      mov    dx, 0200h    ; Row 2, column 0  ; 07/03/2015
000005D2 E871170000                    call   _set_cpos    ; 24/01/2016
                                       ;
                                       ; 06/11/2014
000005D7 E8782C0000                    call   memory_info
                                       ; 14/08/2015
                                       ;call getch ; 28/02/2015
                                 drv_init:
000005DC FB                            sti    ; Enable Interrupts
                                       ; 06/02/2015
000005DD 8B15[F85C0000]                mov    edx, [hd0_type] ; hd0, hd1, hd2, hd3
000005E3 668B1D[F65C0000]              mov    bx, [fd0_type] ; fd0, fd1
                                       ; 22/02/2015
000005EA 6621DB                        and    bx, bx
000005ED 751C                          jnz    short di1
                                       ;
000005EF 09D2                          or     edx, edx
000005F1 752A                          jnz    short di2
                                       ;
                                 setup_error:
000005F3 BE[56190100]                  mov    esi, setup_error_msg
                                 psem:
000005F8 AC                            lodsb
000005F9 08C0                          or     al, al
                                       ;jz    short haltx ; 22/02/2015
000005FB 7427                          jz     short di3
000005FD 56                            push   esi
                                       ; 13/05/2016
000005FE BB07000000                    mov    ebx, 7 ; Black background,
                                                    ; light gray forecolor
                                                    ; Video page 0 (BH=0)
00000603 E8AA160000                    call   _write_tty
00000608 5E                            pop    esi
00000609 EBED                          jmp    short psem

                                 di1:
                                       ; supress 'jmp short T6'
                                       ;  (activate fdc motor control code)
0000060B 66C705[EB060000]90-           mov    word [T5], 9090h ; nop
00000613 90
                                       ;
                                       ;mov   ax, int_0Eh  ; IRQ 6 handler
                                       ;mov   di, 0Eh*4    ; IRQ 6 vector
                                       ;stosw
                                       ;mov   ax, cs
                                       ;stosw
                                       ;; 16/02/2015
                                         ;;mov     dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
                                       ;
00000614 E8AF3B0000                    CALL   DSKETTE_SETUP; Initialize Floppy Disks
                                       ;
00000619 09D2                          or     edx, edx
0000061B 7407                            jz      short di3
                                 di2:
0000061D E8EC3B0000                    call         DISK_SETUP    ; Initialize Fixed Disks
00000622 72CF                            jc      short setup_error
                                 di3:
00000624 E8FF2B0000                    call   setup_rtc_int; 22/05/2015 (dsectrpm.s)
                                       ;
00000629 E8BE110100                    call   display_disks ; 07/03/2015  (Temporary)
                                 ;haltx:
                                       ; 14/08/2015
                                       ;call  getch ; 22/02/2015
                                       ;sti   ; Enable interrupts (for CPU)
                                 ;     ; 29/01/2016
                                 ;     sub    ah, ah ;  read time count
                                 ;     call   int1Ah
                                 ;     mov    edx, ecx ; 18.2 * seconds
                                 ;md_info_msg_wait1:
                                 ;     ; 29/01/2016
                                 ;     mov    ah, 1
                                 ;     call   int16h
                                 ;     jz     short md_info_msg_wait2
                                 ;     xor    ah, ah ; 0
                                 ;       call    int16h
                                 ;     jmp    short md_info_msg_ok
                                 ;md_info_msg_wait2:
                                 ;     sub    ah, ah  ; read time count
                                 ;     call   int1Ah
                                 ;     cmp    edx, ecx ; ; 18.2 * seconds
                                 ;     jna    short md_info_msg_wait3
                                 ;     xchg   edx, ecx
                                 ;md_info_msg_wait3:
                                 ;     sub    ecx, edx
                                 ;     cmp    ecx, 127 ; 7 seconds (18.2 * 7)
                                 ;     jb     short md_info_msg_wait1
                                 ;md_info_msg_ok:
                                       ; 08/09/2016
0000062E 0F20C0                        mov    eax, cr0
00000631 A810                          test   al, 10h  ; Bit 4, ET (Extension Type)
00000633 7408                          jz     short sysinit
                                       ; 27/02/2017
00000635 FE05[F8650100]                inc    byte [fpready]
                                       ; 80387 (FPU) is ready
0000063B DBE3                          fninit ; Initialize Floating-Point Unit
                                 sysinit:
                                       ; 30/06/2015
```

```
0000063D E80C5C0000                    call   sys_init
                                       ;
                                       ;jmp   cpu_reset ; 22/02/2015
                                 hang:
                                       ; 23/02/2015
                                       ;sti                 ; Enable interrupts
00000642 F4                            hlt
                                       ;
                                       ;nop
                                       ;; 03/12/2014
                                       ;; 28/08/2014
                                       ;mov   ah, 11h
                                       ;call  getc
                                       ;jz      _c8
                                       ;
                                       ; 23/02/2015
                                       ; 06/02/2015
                                       ; 07/09/2014
00000643 31DB                          xor    ebx, ebx
00000645 8A1D[66580100]                mov    bl, [ptty]   ; active_page
0000064B 89DE                          mov    esi, ebx
0000064D 66D1E6                        shl    si, 1
00000650 81C6[68580100]                add    esi, ttychr
00000656 668B06                        mov    ax, [esi]
00000659 6621C0                        and    ax, ax
                                       ;jz    short _c8
0000065C 74E4                          jz     short hang
0000065E 66C7060000                    mov    word [esi], 0
00000663 80FB03                        cmp    bl, 3        ; Video page 3
                                       ;jb    short _c8
00000666 72DA                          jb     short hang
                                       ;
                                       ; 13/05/2016
                                       ; 07/09/2014
                                 nxtl:
00000668 6653                          push   bx
0000066A 66BB0E00                      mov    bx, 0Eh      ; Yellow character
                                                           ; on black background
                                                           ; bh = 0 (video page 0)
                                                           ; Retro UNIX 386 v1 - Video Mode 0
                                                           ; (PC/AT Video Mode 3 - 80x25 Alpha.)
0000066E 6650                          push   ax
00000670 E83D160000                    call   _write_tty
00000675 6658                          pop    ax
00000677 665B                          pop    bx
00000679 3C0D                          cmp    al, 0Dh             ; carriage return (enter)
                                       ;jne   short _c8
0000067B 75C5                          jne    short hang
0000067D B00A                          mov    al, 0Ah             ; next line
0000067F EBE7                          jmp    short nxtl

                                 ;_c8:
                                 ;     ; 25/08/2014
                                 ;     cli                        ; Disable interrupts
                                 ;     mov    al, [scounter + 1]
                                 ;     and    al, al
                                 ;     jnz    hang
                                 ;     call   rtc_p
                                 ;     jmp     hang


                                       ; 27/08/2014
                                       ; 20/08/2014
                                 printk:
                                         ;mov    edi, [scr_row]
                                 pkl:
00000681 AC                            lodsb
00000682 08C0                          or     al, al
00000684 7404                          jz     short pkr
00000686 66AB                          stosw
00000688 EBF7                          jmp    short pkl
                                 pkr:
0000068A C3                            retn

                                 ; 28/02/2017
                                 ; 22/01/2017
                                 ; 15/01/2017
                                 ; 14/01/2017
                                 ; 02/01/2017
                                 ; 25/12/2016
                                 ; 19/12/2016
                                 ; 10/12/2016 (callback)
                                 ; 06/06/2016
                                 ; 23/05/2016
                                 ; 22/05/2016 - TRDOS 386 (TRDOS v2.0) Timer Event Modifications
                                 ; 25/07/2015
                                 ; 14/05/2015 (multi tasking -time sharing- 'clock', x_timer)
                                 ; 17/02/2015
                                 ; 06/02/2015 (unix386.s)
                                 ; 11/12/2014 - 22/12/2014 (dsectrm2.s)
                                 ;
                                 ; IBM PC-XT Model 286 Source Code - BIOS2.ASM (06/10/85)
                                 ;
                                 ;-- HARDWARE INT  08 H - ( IRQ LEVEL 0 ) ---------------------------------------
                                 ;     THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF        :
                                 ;     THE 8254 TIMER.  INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR        :
                                 ;     IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND.     :
                                 ;                                                                :
                                 ;     THE INTERRUPT HANDLER MAINTAINS A COUNT (40:6C) OF INTERRUPTS SINCE    :
                                 ;     POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.        :
                                 ;     THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT (40:40)  :
                                 ;     OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE           :
                                 ;     DISKETTE MOTOR(s), AND RESET THE MOTOR RUNNING FLAGS.                   :
```

```
                                 ;     THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH           :
                                 ;     INTERRUPT 1CH AT EVERY TIME TICK.  THE USER MUST CODE A          :
                                 ;     ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.       :
                                 ;-------------------------------------------------------------------------------
                                 ;

                                 timer_int:  ; IRQ 0
                                 ;int_08h:   ; Timer
                                       ; 14/10/2015
                                       ; Here, we are simulating system call entry (for task switch)
                                       ; (If multitasking is enabled,
                                       ; 'clock' procedure may jump to 'sysrelease')

0000068B 1E                            push  ds
0000068C 06                            push  es
0000068D 0FA0                          push  fs
0000068F 0FA8                          push  gs

00000691 60                            pushad ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
00000692 66B91000                      mov     cx, KDATA
00000696 8ED9                            mov     ds, cx
00000698 8EC1                            mov     es, cx
0000069A 8EE1                            mov     fs, cx
0000069C 8EE9                            mov     gs, cx

0000069E 0F20D9                        mov   ecx, cr3
000006A1 890D[5C040300]                mov   [cr3reg], ecx ; save current cr3 register value/content

                                       ; 14/01/2017
000006A7 3B0D[38580100]                cmp   ecx, [k_page_dir]
000006AD 7409                          je    short T3

000006AF 8B0D[38580100]                mov   ecx, [k_page_dir]
000006B5 0F22D9                        mov   cr3, ecx
                                 T3:
                                       ;sti                      ; INTERRUPTS BACK ON
000006B8 66FF05[B8580100]              INC   word [TIMER_LOW]    ; INCREMENT TIME
000006BF 7507                          JNZ   short T4            ; GO TO TEST_DAY
000006C1 66FF05[BA580100]              INC   word [TIMER_HIGH]   ; INCREMENT HIGH WORD OF TIME
                                 T4:                             ; TEST_DAY
000006C8 66833D[BA580100]18            CMP   word [TIMER_HIGH],018H     ; TEST FOR COUNT EQUALING 24 HOURS
000006D0 7519                          JNZ   short T5            ; GO TO DISKETTE_CTL
000006D2 66813D[B8580100]B0-           CMP   word [TIMER_LOW],0B0H
000006DA 00
000006DB 750E                          JNZ   short T5            ; GO TO DISKETTE_CTL

                                 ;-----      TIMER HAS GONE 24 HOURS
                                       ;;SUB AX,AX
                                       ;MOV  [TIMER_HIGH],AX
                                       ;MOV  [TIMER_LOW],AX
000006DD 29C0                          sub   eax, eax
000006DF A3[B8580100]                  mov   [TIMER_LH], eax
                                       ;
000006E4 C605[BC580100]01              MOV   byte [TIMER_OFL],1

                                 ;-----      TEST FOR DISKETTE TIME OUT

                                 T5:
                                       ; 23/12/2014
000006EB EB1D                          jmp   short T6            ; will be replaced with nop, nop
                                                                 ; (9090h) if a floppy disk
                                                                 ; is detected.
                                       ;mov  al,[CS:MOTOR_COUNT]
000006ED A0[BF580100]                  mov   al, [MOTOR_COUNT]
000006F2 FEC8                          dec   al
                                       ;mov  [CS:MOTOR_COUNT], al      ; DECREMENT DISKETTE MOTOR CONTROL
000006F4 A2[BF580100]                  mov   [MOTOR_COUNT], al
                                       ;mov  [ORG_MOTOR_COUNT], al
000006F9 750F                          JNZ   short T6            ; RETURN IF COUNT NOT OUT
000006FB B0F0                          mov   al,0F0h
                                       ;AND  [CS:MOTOR_STATUS],al      ; TURN OFF MOTOR RUNNING BITS
000006FD 2005[BE580100]                and   [MOTOR_STATUS], al
                                       ;and  [ORG_MOTOR_STATUS], al
00000703 B00C                          MOV   AL,0CH              ; bit 3 = enable IRQ & DMA,
                                                                 ; bit 2 = enable controller
                                                                 ;      1 = normal operation
                                                                 ;      0 = reset
                                                                 ; bit 0, 1 = drive select
                                                                 ; bit 4-7 = motor running bits
00000705 66BAF203                      MOV   DX,03F2H            ; FDC CTL PORT
00000709 EE                            OUT   DX,AL               ; TURN OFF THE MOTOR
                                 T6:
                                       ;inc  word [CS:wait_count]      ; 22/12/2014 (byte -> word)
                                                                 ; TIMER TICK INTERRUPT
                                       ;;inc word [wait_count] ;;27/02/2015
                                       ;INT  1CH                 ; TRANSFER CONTROL TO A USER ROUTINE
                                       ;cli
0000070A E857040000                    call  u_timer                   ; TRANSFER CONTROL TO A USER ROUTINE
                                       ; 23/05/2016
0000070F E823F20000                    call  clock               ; Multi Tasking control procedure
                                 T7:
                                       ; 14/10/2015
00000714 B020                          MOV   AL,EOI              ; GET END OF INTERRUPT MASK
00000716 FA                            CLI                       ; DISABLE INTERRUPTS TILL STACK CLEARED
00000717 E620                          OUT   INTA00,AL           ; END OF INTERRUPT TO 8259 - 1
                                       ;
                                 rtc_int_2:
                                       ; 26/12/2016
                                       ;mov  ecx, [cr3reg]
                                       ; 13/01/2017
00000719 803D[D4030300]00              cmp   byte [u.t_lock], 0        ; T_LOCK
00000720 7730                          ja    short timer_int_return  ; Timer Lock : 'sysrele' is needed !
                                       ; 28/02/2017
                                       ; We need to exit if the user's IRQ callback service is in progress!
                                       ; (To prevent a conflict!)
00000722 803D[D8030300]00              cmp    byte [u.r_lock], 0  ; R_LOCK, IRQ callback service lock !
00000729 7727                          ja     short timer_int_return  ; Timer Lock : 'sysrele' is needed !
                                       ; 15/01/2017
0000072B 803D[CC650100]02              cmp    byte [priority], 2
00000732 733A                          jnb    short T8  ; current process has a timer event (15/01/2017)
                                       ; 22/05/2016
00000734 803D[CD650100]00              cmp    byte [p_change], 0 ; in 'set_run_sequence', in 'rtc_p'
0000073B 7615                          jna    short timer_int_return ; 23/05/2016

                                       ; 15/01/2017

                                       ; present process must be changed with high priority process
                                       ;xor   al, al
0000073D 31C0                          xor    eax, eax ; 26/12/2016
0000073F A2[CD650100]                  mov    [p_change], al ; 0
                                       ;mov   byte [priority], 2 ; 15/01/2017 (there is a timer event)

00000744 803D[5B030300]FF              cmp     byte [sysflg], 0FFh ; user or system space ?
0000074B 7416                          je     short rtc_int_3     ; user space ([sysflg]= 0FFh)

                                       ; system space, wait for 'sysret'
                                       ; to change running process
                                       ; with high priority (event) process

0000074D A2[A8030300]                  mov    [u.quant], al ; 0

                                 timer_int_return: ; 23/05/2016 - jump from 'rtc_int' ('rtc_int_2')
00000752 8B0D[5C040300]                mov    ecx, [cr3reg]       ; previous value/content of cr3 register
00000758 0F22D9                        mov    cr3, ecx     ; restore cr3 register content
                                       ;
0000075B 61                            popad ; edi, esi, ebp, temp (icrement esp by 4), ebx, edx, ecx, eax
                                       ;
0000075C 0FA9                          pop    gs
0000075E 0FA1                          pop    fs
00000760 07                            pop    es
00000761 1F                            pop    ds
                                       ;
00000762 CF                            iretd  ; return from interrupt

                                 rtc_int_3:
00000763 FE05[5B030300]                inc    byte [sysflg]       ; now, we are in system space
                                       ;
00000769 E990BF0000                      jmp     sysrelease ; change running process immediatelly

                                 T8:
                                       ; 13/01/2017 (eax -> ebx)
                                       ; callback checking... (19/12/2016)
0000076E 31DB                          xor    ebx, ebx
00000770 871D[D0030300]                xchg   ebx, [u.tcb] ; callback address (0 = normal return)
00000776 09DB                          or     ebx, ebx
00000778 74D8                          jz     short timer_int_return

                                       ; Set user's callback routine as return address from this interrupt
                                       ; and set normal return address as return address from callback
                                       ; routine!!! (19/12/2016)

                                       ; 14/01/2017
                                       ; 13/01/2017 - Timer Lock (T_LOCK)
0000077A FE05[D4030300]                inc    byte [u.t_lock]
00000780 8A0D[5B030300]                mov    cl, [sysflg]
00000786 880D[D5030300]                mov    [u.t_mode], cl

0000078C 8B2D[D4570100]                mov    ebp, [tss.esp0] ; kernel stack address (for ring 0)
00000792 83ED14                        sub    ebp, 20             ; eip, cs, eflags, esp, ss
00000795 892D[5C030300]                mov    [u.sp], ebp
0000079B 8925[60030300]                mov    [u.usp], esp

                                       ;or    word [ebp+8], 200h ; 22/01/2017, force enabling interrupts

000007A1 8B44241C                      mov    eax, [esp+28] ; pushed eax
000007A5 A3[64030300]                  mov    [u.r0], eax

000007AA E87EDE0000                    call   wswap ; save user's registers & status

                                       ; software int is in ring 0 but timer int must return to ring 3
                                       ; so, ring 3 return address and stack registers
                                       ; (eip, cs, eflags, esp, ss)
                                       ; must be copied to timer int return
                                       ; eip will be replaced by callback service routine address

000007AF C605[5B030300]FF              mov    byte [sysflg], 0FFh ; user mode

                                       ; system mode (system call)
                                       ;mov   ebp, [u.sp] ; EIP (u), CS (UCODE), EFLAGS (u),
                                                         ; ESP (u), SS (UDATA)

000007B6 8B4510                        mov    eax, [ebp+16]; SS (UDATA
000007B9 89E6                          mov    esi, esp
000007BB 50                            push   eax
000007BC 50                            push   eax
000007BD 89E7                          mov    edi, esp
000007BF 893D[60030300]                mov    [u.usp], edi
000007C5 B908000000                    mov    ecx, ((ESPACE/4) - 4) ; except DS, ES, FS, GS
000007CA F3A5                          rep    movsd
000007CC B104                          mov    cl, 4
000007CE F3AB                          rep    stosd
000007D0 893D[5C030300]                mov    [u.sp], edi
000007D6 89EE                          mov    esi, ebp
000007D8 B105                          mov    cl, 5 ; EIP (u), CS (UCODE), EFLAGS (u), ESP (u), SS (UDATA)
000007DA F3A5                          rep    movsd

```

```
000007DC 8B0D[B8030300]                mov   ecx, [u.pgdir]
000007E2 890D[5C040300]                mov   [cr3reg], ecx

                                       ; 13/01/207 (eax -> ebx)
                                       ; EBX = callback routine address (virtual, not physical address!)

                                       ; 09/01/2017
                                       ; !!! CALLBACK ROUTINE MUST BE ENDED/RETURNED WITH 'sysrele'
                                       ;     system call !!!
                                       ; 25/12/2016
                                       ; Callback Note: (19/12/2016)
                                       ; !!! CALLBACK ROUTINE MUST BE ENDED/RETURNED WITH 'RETN' !!!
                                       ;     pushf ; save flags
                                       ;     <callback service code>
                                       ;     popf  ; restore flags
                                       ;     retn ; return to normal running address
                                       ;

                                       ; 15/01/2017
                                       ; 14/01/2017
                                       ; 13/01/2017 (eax -> ebx)
                                       ; 10/01/2017
                                 set_callback_addr:
                                       ; 09/01/2017 (**)
                                       ; 02/01/2017 (*)
                                       ; 25/12/2016 (*)
                                       ; 19/12/2016 (TRDOS 386 feature only!)
                                       ;
                                       ; This routine sets return address
                                       ; to start of user's interrupt
                                       ; service (callback) address
                                       ;; and sets callback 'retn' address to normal
                                       ;; return address of user's running code!
                                       ;
                                       ; INPUT:
                                       ;     EBX = callback routine/service address
                                       ;           (virtual, not physical address!)
                                       ;     [u.sp] = kernel stack, points to
                                       ;             user's EIP,CS,EFLAGS,ESP,SS
                                       ;             registers.
                                       ; OUTPUT:
                                       ;     EIP (user) = callback (service) address
                                       ;     CS (user) = UCODE
                                       ;     EFLAGS (user) = flags before callback
                                       ;       ESP (user) = ESP-4 (user, before callback)
                                       ;     [ESP](user) = EIP (user) before callback
                                       ;
                                       ; Note: If CPU was in user mode while entering
                                       ;     the timer interrupt service routine,
                                       ;     'IRET' will get return to callback routine
                                       ;     immediately. If CPU was in system/kernel mode
                                       ;     'iret' will get return to system call and
                                       ;     then, callback routine will be return address
                                       ;     from system call. (User's callback/service code
                                       ;     will be able to return to normal return address
                                       ;     via an 'retn' at the end.)
                                       ;
                                       ; Note(**): User's callback service code must be ended
                                       ;     with a 'sysrele' sytstem call ! (09/01/2017)
                                       ;
                                       ;     For example:
                                       ;
                                       ;     timer_callback:
                                       ;         ...
                                       ;         inc      dword [time_counter]
                                       ;         ...
                                       ;         mov eax, 39 ; 'sysrele'
                                       ;         int 40h ; TRDOS 386 system call (interrupt)
                                       ;
                                       ;
                                       ;; Note(*): User's callback service code must preserve cpu
                                       ;;    flags if it has any instructions which changes
                                       ;;    flags in the service code. (25/12/2016)
                                       ;;
                                       ;;    For example:
                                       ;;
                                       ;;    timer_callback:
                                       ;;        pushf ; save flags
                                       ;;        ; this instruction changes zero flag
                                       ;;        inc      dword [time_counter]
                                       ;;        popf ; restore flags
                                       ;;        retn ; return to normal user code
                                       ;;             (which is interrupted by the
                                       ;;              timer interput)
                                       ;;

                                       ; 15/01/2017
000007E8 8B2D[5C030300]                mov   ebp, [u.sp]; kernel's stack, points to EIP (user)
000007EE 895D00                        mov   [ebp], ebx
000007F1 E95CFFFFFF                    jmp   timer_int_return

                                       ; 15/01/2017
                                       ; 13/01/2017
                                       ; 19/12/2016
                                       ; 06/06/2016
                                       ; 23/05/2016
                                       ; 22/05/2016
                                       ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
                                       ; 26/02/2015
                                       ; 07/09/2014
                                       ; 25/08/2014
                                 rtc_int:       ; Real Time Clock Interrupt (IRQ 8)
                                       ; 22/05/2016
```

```
000007F6 1E                            push   ds ; ** ; 23/05/2016
000007F7 50                            push   eax ; *
000007F8 66B81000                      mov    ax, KDATA
000007FC 8ED8                          mov    ds, ax
                                       ;
000007FE 8A25[B6580100]                mov    ah, [RTC_2Hz] ;  2 Hz interrupt to 1 Hz function
00000804 80F401                        xor    ah, 1
00000807 8825[B6580100]                mov    [RTC_2Hz], ah ; 1 = 0.5 second, 0 = 1 second
0000080D 753B                          jnz    short rtc_int_return ; half second
                                       ; 1 second
                                 rtc_int_0:
                                       ; 22/05/2016
0000080F 58                            pop    eax ; *
                                       ;
                                       ; 14/10/2015 ('timer_int')
                                       ; Here, we are simulating system call entry (for task switch)
                                       ; (If multitasking is enabled,
                                       ; 'clock' procedure may jump to 'sysrelease')
                                       ;push  ds ; ** ; 23/05/2016
00000810 06                            push   es
00000811 0FA0                          push   fs
00000813 0FA8                          push   gs
00000815 60                            pushad  ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
00000816 66B91000                      mov     cx, KDATA
                                         ;mov    ds, cx ; 06/06/2016
0000081A 8EC1                            mov     es, cx
0000081C 8EE1                            mov     fs, cx
0000081E 8EE9                            mov     gs, cx
                                       ;
00000820 0F20D9                        mov    ecx, cr3
00000823 890D[5C040300]                mov    [cr3reg], ecx ; save current cr3 register value/content
                                       ;
00000829 803D[D4030300]00              cmp    byte [u.t_lock], 0 ; timer lock (callback) status ?
00000830 7711                          ja     short rtc_int_1       ; yes

                                       ; 15/01/2017
00000832 3B0D[38580100]                cmp    ecx, [k_page_dir]
00000838 7409                          je     short rtc_int_1

0000083A 8B0D[38580100]                mov    ecx, [k_page_dir]
00000840 0F22D9                        mov    cr3, ecx
                                 rtc_int_1:
                                       ; Timer event (kernel) functions must be performed with
                                       ; 1 second intervals - TRDOS 386 (TRDOS v2.0) feature ! -
                                       ;
                                       ; 25/08/2014
00000843 E81A030000                    call   rtc_p  ; 19/05/2016 - major modification

                                       ; 23/05/2016
00000848 28E4                          sub    ah, ah ; 0
                                       ; 22/05/2016 - TRDOS 386 timer event modifications
                                 rtc_int_return: ; 19/05/2016
                                       ; 22/02/2015 - dsectpm.s
                                       ; [ source: http://wiki.osdev.org/RTC ]
                                       ; read status register C to complete procedure
                                       ;(it is needed to get a next IRQ 8)
0000084A B00C                          mov    al, 0Ch ;
0000084C E670                          out    70h, al ; select register C
0000084E 90                            nop
0000084F E471                          in     al, 71h ; just throw away contents
                                       ; 22/02/2015
00000851 B020                          MOV    AL,EOI       ; END OF INTERRUPT
                                       ;CLI                ; DISABLE INTERRUPTS TILL STACK CLEARED
00000853 E6A0                          OUT    INTB00,AL    ; FOR CONTROLLER #2

                                       ; 23/05/2016
00000855 B020                          MOV    AL,EOI       ; GET END OF INTERRUPT MASK
00000857 FA                            CLI                 ; DISABLE INTERRUPTS TILL STACK CLEARED
00000858 E620                          OUT    INTA00,AL    ; END OF INTERRUPT TO 8259 - 1
                                       ;
                                       ; 23/05/2016
0000085A 20E4                          and    ah, ah
0000085C 0F84B7FEFFFF                    jz      rtc_int_2

                                       ; ah = 1 (half second)
00000862 58                            pop    eax ; *
00000863 1F                            pop    ds  ; **
00000864 CF                            iretd

                                 ; ///////////////

                                       ; 28/08/2014
                                 irq0:
00000865 6A00                            push       dword 0
00000867 EB48                          jmp    short which_irq
                                 irq1:
00000869 6A01                            push       dword 1
0000086B EB44                          jmp    short which_irq
                                 irq2:
0000086D 6A02                            push       dword 2
0000086F EB40                          jmp    short which_irq
                                 irq3:
                                       ; 20/11/2015
                                       ; 24/10/2015
00000871 2EFF15[F5FB0000]              call   dword [cs:com2_irq3]
00000878 6A03                          push   dword 3
0000087A EB35                          jmp    short which_irq
                                 irq4:
                                       ; 20/11/2015
                                       ; 24/10/2015
0000087C 2EFF15[F1FB0000]              call   dword [cs:com1_irq4]
00000883 6A04                            push       dword 4
00000885 EB2A                          jmp    short which_irq
```

```
                                 irq5:
00000887 6A05                            push       dword 5
00000889 EB26                          jmp    short which_irq
                                 irq6:
0000088B 6A06                            push       dword 6
0000088D EB22                          jmp    short which_irq
                                 irq7:
0000088F 6A07                            push       dword 7
00000891 EB1E                          jmp    short which_irq
                                 irq8:
00000893 6A08                            push       dword 8
00000895 EB1A                          jmp    short which_irq
                                 irq9:
00000897 6A09                            push       dword 9
00000899 EB16                          jmp    short which_irq
                                 irq10:
0000089B 6A0A                            push       dword 10
0000089D EB12                          jmp    short which_irq
                                 irq11:
0000089F 6A0B                            push       dword 11
000008A1 EB0E                          jmp    short which_irq
                                 irq12:
000008A3 6A0C                            push       dword 12
000008A5 EB0A                          jmp    short which_irq
                                 irq13:
000008A7 6A0D                            push       dword 13
000008A9 EB06                          jmp    short which_irq
                                 irq14:
000008AB 6A0E                            push       dword 14
000008AD EB02                          jmp    short which_irq
                                 irq15:
000008AF 6A0F                            push       dword 15
                                       ;jmp   short which_irq

                                       ; 22/01/2017
                                       ; 19/10/2015
                                       ; 29/08/2014
                                       ; 21/08/2014
                                 which_irq:
000008B1 870424                        xchg   eax, [esp]  ; 28/08/2014
000008B4 53                            push   ebx
000008B5 56                            push   esi
000008B6 57                            push   edi
000008B7 1E                            push   ds
000008B8 06                            push   es
                                       ;
000008B9 88C3                          mov    bl, al
                                       ;
000008BB B810000000                    mov    eax, KDATA
000008C0 8ED8                          mov    ds, ax
000008C2 8EC0                          mov    es, ax
                                       ; 19/10/2015
000008C4 FC                            cld
                                         ; 27/08/2014
000008C5 8105[48160100]A000-             add     dword [scr_row], 0A0h
000008CD 0000
                                       ;
000008CF B417                          mov    ah, 17h      ; blue (1) background,
                                                    ; light gray (7) forecolor
000008D1 8B3D[48160100]                  mov     edi, [scr_row]
000008D7 B049                          mov    al, 'I'
000008D9 66AB                          stosw
000008DB B052                          mov    al, 'R'
000008DD 66AB                          stosw
000008DF B051                          mov    al, 'Q'
000008E1 66AB                          stosw
000008E3 B020                          mov    al, ' '
000008E5 66AB                          stosw
000008E7 88D8                          mov    al, bl
000008E9 3C0A                          cmp    al, 10
000008EB 7208                          jb     short ii1
000008ED B031                          mov    al, '1'
000008EF 66AB                          stosw
000008F1 88D8                          mov    al, bl
000008F3 2C0A                          sub    al, 10
                                 ii1:
000008F5 0430                          add    al, '0'
000008F7 66AB                          stosw
000008F9 B020                          mov    al, ' '
000008FB 66AB                          stosw
000008FD B021                          mov    al, '!'
000008FF 66AB                          stosw
00000901 B020                          mov    al, ' '
00000903 66AB                          stosw
                                       ; 23/02/2015
00000905 80FB07                        cmp    bl, 7 ; check for IRQ 8 to IRQ 15
00000908 7604                          jna    ii2
                                       ; 22/01/2017
0000090A B020                          mov    al, 20h  ; END OF INTERRUPT COMMAND TO
0000090C E6A0                          out    0A0h, al ; the 2nd 8259
                                 ii2:
0000090E B020                          mov    al, 20h  ; END OF INTERRUPT COMMAND TO
00000910 E620                          out    20h, al ; the 2nd 8259
00000912 E9CD010000                    jmp     iiret
                                       ;
                                       ; 22/08/2014
                                       ;mov   al, 20h ; END OF INTERRUPT COMMAND TO 8259
                                       ;out   20h, al      ; 8259 PORT
                                       ;
                                       ;pop   es
                                       ;pop   ds
                                       ;pop   edi
                                       ;pop   esi
                                       ;pop   ebx
                                       ;pop   eax
                                       ;iret

                                       ; 02/04/2015
                                       ; 25/08/2014
                                 exc0:
00000917 6A00                            push       dword 0
00000919 E990000000                      jmp     cpu_except
                                 exc1:
0000091E 6A01                            push       dword 1
00000920 E989000000                      jmp     cpu_except
                                 exc2:
00000925 6A02                            push       dword 2
00000927 E982000000                      jmp     cpu_except
                                 exc3:
0000092C 6A03                            push       dword 3
0000092E EB7E                            jmp     cpu_except
                                 exc4:
00000930 6A04                            push       dword 4
00000932 EB7A                            jmp     cpu_except
                                 exc5:
00000934 6A05                            push       dword 5
00000936 EB76                            jmp     cpu_except
                                 exc6:
00000938 6A06                            push       dword 6
0000093A EB72                            jmp     cpu_except
                                 exc7:
0000093C 6A07                            push       dword 7
0000093E EB6E                            jmp     cpu_except
                                 exc8:
                                       ; [esp] = Error code
00000940 6A08                            push       dword 8
00000942 EB5C                            jmp     cpu_except_en
                                 exc9:
00000944 6A09                            push       dword 9
00000946 EB66                            jmp     cpu_except
                                 exc10:
                                       ; [esp] = Error code
00000948 6A0A                            push       dword 10
0000094A EB54                            jmp     cpu_except_en
                                 exc11:
                                       ; [esp] = Error code
0000094C 6A0B                            push       dword 11
0000094E EB50                            jmp     cpu_except_en
                                 exc12:
                                       ; [esp] = Error code
00000950 6A0C                            push       dword 12
00000952 EB4C                            jmp     cpu_except_en
                                 exc13:
                                       ; [esp] = Error code
00000954 6A0D                            push       dword 13
00000956 EB48                            jmp     cpu_except_en
                                 exc14:
                                       ; [esp] = Error code
00000958 6A0E                            push       dword 14
0000095A EB44                          jmp    short cpu_except_en
                                 exc15:
0000095C 6A0F                            push       dword 15
0000095E EB4E                            jmp     cpu_except
                                 exc16:
00000960 6A10                            push       dword 16
00000962 EB4A                            jmp     cpu_except
                                 exc17:
                                       ; [esp] = Error code
00000964 6A11                            push       dword 17
00000966 EB38                          jmp    short cpu_except_en
                                 exc18:
00000968 6A12                            push       dword 18
0000096A EB42                          jmp    short cpu_except
                                 exc19:
0000096C 6A13                            push       dword 19
0000096E EB3E                          jmp    short cpu_except
                                 exc20:
00000970 6A14                            push       dword 20
00000972 EB3A                          jmp    short cpu_except
                                 exc21:
00000974 6A15                            push       dword 21
00000976 EB36                          jmp    short cpu_except
                                 exc22:
00000978 6A16                            push       dword 22
0000097A EB32                          jmp    short cpu_except
                                 exc23:
0000097C 6A17                            push       dword 23
0000097E EB2E                          jmp    short cpu_except
                                 exc24:
00000980 6A18                            push       dword 24
00000982 EB2A                          jmp    short cpu_except
                                 exc25:
00000984 6A19                            push       dword 25
00000986 EB26                          jmp    short cpu_except
                                 exc26:
00000988 6A1A                            push       dword 26
0000098A EB22                          jmp    short cpu_except
                                 exc27:
0000098C 6A1B                            push       dword 27
0000098E EB1E                          jmp    short cpu_except
                                 exc28:
00000990 6A1C                            push       dword 28
00000992 EB1A                          jmp    short cpu_except
                                 exc29:
00000994 6A1D                            push       dword 29
00000996 EB16                          jmp    short cpu_except
```

```
                                 exc30:
00000998 6A1E                            push       dword 30
0000099A EB04                          jmp   short cpu_except_en
                                 exc31:
0000099C 6A1F                            push       dword 31
0000099E EB0E                            jmp     short cpu_except

                                       ; 19/10/2015
                                       ; 19/09/2015
                                       ; 01/09/2015
                                       ; 28/08/2015
                                       ; 28/08/2014
                                 cpu_except_en:
000009A0 87442404                      xchg  eax, [esp+4] ; Error code
000009A4 36A3[78050300]                mov   [ss:error_code], eax
000009AA 58                            pop   eax  ; Exception number
000009AB 870424                        xchg  eax, [esp]
                                             ; eax = eax before exception
                                             ; [esp] -> exception number
                                             ; [esp+4] -> EIP to return
                                       ; 22/01/2017
                                       ; 19/10/2015
                                       ; 19/09/2015
                                       ; 01/09/2015
                                       ; 28/08/2015
                                       ; 29/08/2014
                                       ; 28/08/2014
                                       ; 25/08/2014
                                       ; 21/08/2014
                                 cpu_except:  ; CPU Exceptions
000009AE FC                            cld
000009AF 870424                        xchg  eax, [esp]
                                             ; eax = Exception number
                                             ; [esp] = eax (before exception)
000009B2 53                            push  ebx
000009B3 56                            push  esi
000009B4 57                            push  edi
000009B5 1E                            push  ds
000009B6 06                            push  es
                                       ; 28/08/2015
000009B7 66BB1000                      mov   bx, KDATA
000009BB 8EDB                          mov   ds, bx
000009BD 8EC3                          mov   es, bx
000009BF 0F20DB                        mov   ebx, cr3
000009C2 53                            push  ebx ; (*) page directory
                                       ; 19/10/2015
000009C3 FC                            cld
                                       ; 25/03/2015
000009C4 8B1D[38580100]                mov   ebx, [k_page_dir]
000009CA 0F22DB                        mov   cr3, ebx
                                       ; 28/08/2015
000009CD 83F80E                        cmp   eax, 0Eh ; 14, PAGE FAULT
000009D0 750F                          jne   short cpu_except_nfp
000009D2 E87B440000                    call  page_fault_handler
000009D7 21C0                          and   eax, eax
000009D9 0F8401010000                    jz  iiretp ; 01/09/2015
000009DF B00E                          mov   al, 0Eh ; 14
                                 cpu_except_nfp:
                                       ; 23/08/2016
000009E1 803D[C25E0000]03              cmp   byte [CRT_MODE], 3
000009E8 7409                          je    short cpu_except_mode_3
000009EA 50                            push  eax
000009EB B003                          mov   al, 3
000009ED E8730B0000                    call  _set_mode
000009F2 58                            pop   eax
                                 cpu_except_mode_3:
                                       ; 02/04/2015
000009F3 BB[42060000]                  mov   ebx, hang
000009F8 875C241C                      xchg  ebx, [esp+28]
                                             ; EIP (points to instruction which faults)
                                             ; New EIP (hang)
000009FC 891D[7C050300]                mov   [FaultOffset], ebx
00000A02 C744242008000000              mov   dword [esp+32], KCODE ; kernel's code segment
00000A0A 814C242400020000              or    dword [esp+36], 200h ; enable interrupts (set IF)
                                       ;
00000A12 88C4                          mov   ah, al
00000A14 240F                          and   al, 0Fh
00000A16 3C09                          cmp   al, 9
00000A18 7602                          jna   short h1ok
00000A1A 0407                          add   al, 'A'-':'
                                 h1ok:
00000A1C C0EC04                        shr   ah, 4
00000A1F 80FC09                        cmp   ah, 9
00000A22 7603                          jna   short h2ok
00000A24 80C407                        add   ah, 'A'-':'
                                 h2ok:
00000A27 86E0                          xchg  ah, al
00000A29 66053030                      add   ax, '00'
00000A2D 66A3[A0180100]                mov   [excnstr], ax
                                       ;
                                       ; 29/08/2014
00000A33 A1[7C050300]                  mov   eax, [FaultOffset]
00000A38 51                            push  ecx
00000A39 52                            push  edx
00000A3A 89E3                          mov   ebx, esp
                                       ; 28/08/2015
00000A3C B910000000                    mov   ecx, 16         ; divisor value to convert binary number
                                                      ; to hexadecimal string
                                       ;mov  ecx, 10           ; divisor to convert
                                                        ; binary number to decimal string
                                 b2d1:
00000A41 31D2                          xor   edx, edx
00000A43 F7F1                          div   ecx
```

```
00000A45 6652                          push   dx
00000A47 39C8                          cmp    eax, ecx
00000A49 73F6                          jnb    short b2d1
00000A4B BF[AB180100]                  mov    edi, EIPstr ; EIP value
                                                        ; points to instruction which faults
                                       ; 28/08/2015
00000A50 89C2                          mov    edx, eax
                                 b2d2:
                                       ;add   al, '0'
00000A52 8A82[1B330000]                mov    al, [edx+hexchrs]
00000A58 AA                            stosb            ; write hexadecimal digit to its place
00000A59 39E3                          cmp    ebx, esp
00000A5B 7606                          jna    short b2d3
00000A5D 6658                          pop    ax
00000A5F 88C2                          mov    dl, al
00000A61 EBEF                          jmp    short b2d2
                                 b2d3:
00000A63 B068                          mov    al, 'h' ; 28/08/2015
00000A65 AA                            stosb
00000A66 B020                          mov    al, 20h          ; space
00000A68 AA                            stosb
00000A69 30C0                          xor    al, al    ; to do it an ASCIIZ string
00000A6B AA                            stosb
                                       ;
00000A6C 5A                            pop    edx
00000A6D 59                            pop    ecx
                                       ;
00000A6E B44F                          mov    ah, 4Fh      ; red (4) background,
                                                    ; white (F) forecolor
00000A70 BE[90180100]                  mov    esi, exc_msg ; message offset
                                       ;
                                       ; 20/01/2017 (!cpu exception!)
                                       ;
00000A75 8105[48160100]A000-             add    dword [scr_row], 0A0h
00000A7D 0000
00000A7F 8B3D[48160100]                  mov    edi, [scr_row]
                                       ;
00000A85 C605[5B030300]00              mov    byte [sysflg], 0  ; system mode
00000A8C FB                              sti
                                       ;
00000A8D E8EFFBFFFF                    call   printk
                                       ;
00000A92 B410                          mov    ah, 10h
00000A94 E87D010000                    call   int16h ; getc
                                       ;
00000A99 B003                          mov    al, 3
00000A9B E8C50A0000                    call   _set_mode
                                       ;
00000AA0 B801000000                    mov    eax, 1
00000AA5 E9BBBD0000                    jmp    sysexit ; terminate process !!!

                                       ; 22/01/2017
                                       ; 18/04/2016
                                       ; 28/08/2015
                                       ; 23/02/2015
                                       ; 20/08/2014
                                 ignore_int:
00000AAA 50                            push   eax
00000AAB 53                            push   ebx ; 23/02/2015
00000AAC 56                            push   esi
00000AAD 57                            push   edi
00000AAE 1E                            push   ds
00000AAF 06                            push   es
                                       ; 18/04/2016
00000AB0 66B81000                      mov    ax, KDATA
00000AB4 8ED8                          mov    ds, ax
00000AB6 8EC0                          mov    es, ax
                                       ; 28/08/2015
00000AB8 0F20D8                        mov    eax, cr3
00000ABB 50                            push   eax ; (*) page directory
                                       ;
00000ABC B467                          mov    ah, 67h      ; brown (6) background,
                                                    ; light gray (7) forecolor
00000ABE BE[58170100]                  mov    esi, int_msg ; message offset
                                 piemsg:
                                         ; 27/08/2014
00000AC3 8105[48160100]A000-             add     dword [scr_row], 0A0h
00000ACB 0000
00000ACD 8B3D[48160100]                  mov     edi, [scr_row]
                                         ;
00000AD3 E8A9FBFFFF                    call   printk
                                       ;
                                       ; 23/02/2015
00000AD8 B020                          mov    al, 20h  ; END OF INTERRUPT COMMAND TO
00000ADA E6A0                          out    0A0h, al ; the 2nd 8259
                                       ; 22/08/2014
00000ADC B020                          mov    al, 20h ; END OF INTERRUPT COMMAND TO 8259
00000ADE E620                          out    20h, al      ; 8259 PORT
                                 iiretp:
                                       ; 22/01/2017
                                       ; 01/09/2015
                                       ; 28/08/2015
00000AE0 58                            pop    eax ; (*) page directory
00000AE1 0F22D8                        mov    cr3, eax
                                 iiret:
00000AE4 07                            pop    es
00000AE5 1F                            pop    ds
00000AE6 5F                            pop    edi
00000AE7 5E                            pop    esi
00000AE8 5B                            pop    ebx ; 29/08/2014
00000AE9 58                            pop    eax
00000AEA CF                            iretd
```

```
                                       ; 23/05/2016
                                       ; 22/08/2014
                                       ; IBM PC/AT BIOS source code ----- 10/06/85 (bios.asm)
                                       ; (INT 1Ah)
                                       ;; Linux (v0.12) source code (main.c) by Linus Torvalds (1991)
                                 time_of_day:
00000AEB E8EE500000                    call   UPD_IPR                    ; WAIT TILL UPDATE NOT IN PROGRESS
00000AF0 726F                            jc      short time_of_day_retn ; 23/05/2016
00000AF2 B000                          mov    al, CMOS_SECONDS
00000AF4 E800510000                    call   CMOS_READ
00000AF9 A2[A8580100]                  mov    [time_seconds], al
00000AFE B002                          mov    al, CMOS_MINUTES
00000B00 E8F4500000                    call   CMOS_READ
00000B05 A2[A9580100]                  mov    [time_minutes], al
00000B0A B004                          mov    al, CMOS_HOURS
00000B0C E8E8500000                    call   CMOS_READ
00000B11 A2[AA580100]                    mov     [time_hours], al
00000B16 B006                          mov    al, CMOS_DAY_WEEK
00000B18 E8DC500000                    call   CMOS_READ
00000B1D A2[AB580100]                  mov    [date_wday], al
00000B22 B007                          mov    al, CMOS_DAY_MONTH
00000B24 E8D0500000                    call   CMOS_READ
00000B29 A2[AC580100]                  mov    [date_day], al
00000B2E B008                          mov    al, CMOS_MONTH
00000B30 E8C4500000                    call   CMOS_READ
00000B35 A2[AD580100]                  mov    [date_month], al
00000B3A B009                          mov    al, CMOS_YEAR
00000B3C E8B8500000                    call   CMOS_READ
00000B41 A2[AE580100]                  mov    [date_year], al
00000B46 B032                          mov    al, CMOS_CENTURY
00000B48 E8AC500000                    call   CMOS_READ
00000B4D A2[AF580100]                  mov    [date_century], al
                                       ;
00000B52 B000                          mov    al, CMOS_SECONDS
00000B54 E8A0500000                    call   CMOS_READ
00000B59 3A05[A8580100]                cmp    al, [time_seconds]
00000B5F 758A                          jne    short time_of_day

                                 time_of_day_retn:
00000B61 C3                            retn

                                       ; 15/01/2017
                                       ; 10/06/2016
                                       ; 07/06/2016
                                       ; 06/06/2016
                                       ; 23/05/2016
                                 rtc_p:
00000B62 B101                          mov    cl, 1 ; 15/01/2017
00000B64 EB02                          jmp    short rtc_p0
                                 u_timer:
                                       ; Timer Events with 18.2 Hz Timer Ticks
                                       ; (and also timer events with RTC seconds)
00000B66 28C9                          sub    cl, cl ; mov cl, 0 ; 15/01/2017
                                 rtc_p0:
                                       ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
                                       ; Major Modification:
                                       ; Check and Perform Timer Events (for RTC)
                                       ; 25/08/2014 - 07/09/2014
                                       ; Retro UNIX 386 v1:
                                       ; Print Real Time Clock content

                                       ; 15/01/2017
00000B68 880D[CC650100]                mov    byte [priority], cl ; 0 or 1 (not 2)
00000B6E 8A2D[CF650100]                mov    ch, [timer_events]
00000B74 20ED                          and    ch, ch
00000B76 7420                          jz     short rtc_p3

00000B78 BE[60040300]                  mov    esi, timer_set  ; beginning address of
                                                         ; timer events space
                                 rtc_p1:
00000B7D 8B06                          mov    eax, [esi]
00000B7F 20C0                          and    al, al ; 0 = free, >0 = process no.
00000B81 7416                          jz     short rtc_p4
                                       ;
00000B83 C1C810                        ror    eax, 16
                                       ; ah = response value, al = interrupt type
                                       ; 15/01/2017
                                       ; cl = interrupt source
                                       ;       1 = RTC, 0 = PIT
00000B86 38C8                          cmp    al, cl
00000B88 750A                          jne    short rtc_p2 ; not as requested or undefined !
00000B8A 3C01                          cmp    al, 1 ; 1 ; RTC interrupt ?
00000B8C 7410                          je     short rtc_p5 ; yes, check for response
                                       ; 06/06/2016 - 18.2 Hz Timer Ticks
00000B8E 836E080A                      sub    dword [esi+8], 10 ; 1 tick = 10
00000B92 7613                          jna    short rtc_p6  ; continue for responding
                                 rtc_p2:
                                       ; 15/01/2017 (cl -> ch)
                                       ; 07/06/2016
00000B94 FECD                          dec    ch    ; remain count of timer events
00000B96 7501                          jnz    short rtc_p4
                                 rtc_p3:
00000B98 C3                            retn
                                 rtc_p4:
                                       ;cmp   esi, timer_set + 240 ; 15*16 (last event)
                                       ;jnb   short rtc_p3 ; end of timer event space
00000B99 83C610                        add    esi, 16 ; next timer event
00000B9C EBDF                          jmp    short rtc_p1
                                 rtc_p5:
                                       ; current timer count ; 06/06/2016 (182)
00000B9E 816E08B6000000                sub    dword [esi+8], 182 ; 1 second (10*18.2)
00000BA5 77ED                          ja     short rtc_p2  ; check for the next
                                 rtc_p6:
                                       ; it is the time of response!
00000BA7 8B5E04                        mov    ebx, [esi+4] ; set (count limit) value
00000BAA 895E08                        mov    [esi+8], ebx ; reset count down value
                                                         ; to count limit
                                       ; 19/12/2016
                                       ; 10/12/2016 - timer callback modification
00000BAD 8B7E0C                        mov    edi, [esi+12] ; response (or callback) address
00000BB0 807E0100                      cmp    byte [esi+1], 0 ; >0 = callback
00000BB4 762A                          jna    short rtc_p8

                                       ; timer callback !
00000BB6 0FB61E                        movzx ebx, byte [esi] ; process number (>0)
00000BB9 89D8                          mov    eax, ebx
00000BBB C0E302                        shl    bl, 2 ; *4
00000BBE 89BB[0C010300]                mov    [ebx+p.tcb-4], edi ; user's callback service addr
00000BC4 3A05[B3030300]                cmp    al, [u.uno]
00000BCA 7521                          jne    short rtc_p9
00000BCC 893D[D0030300]                mov    [u.tcb], edi
                                 rtc_p7:
                                       ; 15/01/2017
00000BD2 B002                          mov    al, 2
00000BD4 A2[CC650100]                  mov    [priority], al ; 2
                                       ; 10/01/2017
                                       ;mov  byte [u.pri], 2
00000BD9 A2[A9030300]                  mov    [u.pri], al ; 2
00000BDE EBB4                          jmp    short rtc_p2
                                 rtc_p8:
                                       ; response address is physical address of
                                       ; the program's response (signal return) byte
                                       ; 06/06/2016
                                       ;mov  edi, [esi+12] ; response address
00000BE0 8827                          mov    [edi], ah    ; response value
                                       ;
00000BE2 C1C010                        rol    eax, 16
                                       ; 15/01/2017
00000BE5 3A05[B3030300]                cmp    al, [u.uno] ; running process ?
00000BEB 74E5                          je     short rtc_p7
                                 rtc_p9:
                                       ; al = process number  ; 10/06/2016
00000BED B202                          mov    dl, 2 ; priority, 2 = event (high)
00000BEF E8F7EC0000                    call   set_run_sequence ; 19/05/2016
00000BF4 EB9E                          jmp    short rtc_p2 ; 10/06/2016


                                 ; Default IRQ 7 handler against spurious IRQs (from master PIC)
                                 ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
                                 default_irq7:
00000BF6 6650                          push   ax
00000BF8 B00B                          mov    al, 0Bh  ; In-Service register
00000BFA E620                          out    20h, al
00000BFC EB00                            jmp short $+2
00000BFE EB00                          jmp short $+2
00000C00 E420                          in     al, 20h
00000C02 2480                          and    al, 80h ; bit 7 (is it real IRQ 7 or fake?)
00000C04 7404                            jz      short irq7_iret ; Fake (spurious) IRQ, do not send EOI
00000C06 B020                            mov     al, 20h ; EOI
00000C08 E620                          out    20h, al
                                 irq7_iret:
00000C0A 6658                          pop    ax
00000C0C CF                            iretd

                                 bcd_to_ascii:
                                       ; 25/08/2014
                                       ; INPUT ->
                                       ;      al = Packed BCD number
                                       ; OUTPUT ->
                                       ;      ax  = ASCII word/number
                                       ;
                                       ; Erdogan Tan - 1998 (proc_hex) - TRDOS.ASM (2004-2011)
                                       ;
00000C0D D410                          db 0D4h,10h                     ; Undocumented inst. AAM
                                                                  ; AH = AL / 10h
                                                                  ; AL = AL MOD 10h
00000C0F 660D3030                      or ax,'00'                      ; Make it ASCII based

00000C13 86E0                            xchg ah, al

00000C15 C3                            retn


                                     %include 'keyboard.s' ; 07/03/2015
                             <1> ; ****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - keyboard.s
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 15/01/2017
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 17/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Turkish Rational DOS
                             <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
                             <1> ;
                             <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                             <1> ; keyboard.inc (17/10/2015)
                             <1> ;
                             <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)
                             <1> ; ****************************************************************************
                             <1>
                             <1> ; Retro UNIX 386 v1 Kernel - KEYBOARD.INC
                             <1> ; Last Modification: 17/10/2015
                             <1> ;               (Keyboard Data is in 'KYBDATA.INC')
                             <1> ;
                             <1> ; ///////// KEYBOARD FUNCTIONS (PROCEDURES) ///////////////
                             <1>
                             <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>
                             <1> ; 03/12/2014
                             <1> ; 26/08/2014
                             <1> ; KEYBOARD I/O
                             <1> ; (INT_16h - Retro UNIX 8086 v1 - U9.ASM, 30/06/2014)
                             <1>
                             <1> ;NOTE: 'k0' to 'k7' are name of OPMASK registers.
                             <1> ;     (The reason of using '_k' labels!!!) (27/08/2014)
                             <1> ;NOTE: 'NOT' keyword is '~' unary operator in NASM.
                             <1> ;     ('NOT LC_HC' --> '~LC_HC') (bit reversing operator)
                             <1>
                             <1> int16h:      ; 30/06/2015
                             <1> ;getc:
00000C16 9C                  <1>       pushfd ; 28/08/2014
00000C17 0E                  <1>       push   cs
00000C18 E801000000          <1>       call   KEYBOARD_IO_1 ; getc_int
00000C1D C3                  <1>       retn
                             <1>
                             <1> getc_int:
                             <1>       ; 28/02/2015
                             <1>       ; 03/12/2014 (derivation from pc-xt-286 bios source code -1986-,
                             <1>       ;            instead of pc-at bios - 1985-)
                             <1>       ; 28/08/2014 (_k1d)
                             <1>       ; 30/06/2014
                             <1>       ; 03/03/2014
                             <1>       ; 28/02/2014
                             <1>       ; Derived from "KEYBOARD_IO_1" procedure of IBM "pc-xt-286"
                             <1>       ; rombios source code (21/04/1986)
                             <1>       ;       'keybd.asm', INT 16H, KEYBOARD_IO
                             <1>       ;
                             <1>       ; KYBD --- 03/06/86  KEYBOARD BIOS
                             <1>       ;
                             <1>       ;--- INT 16 H ------------------------------------------------------------------
                             <1>       ; KEYBOARD I/O                                                           :
                             <1>       ;      THESE ROUTINES PROVIDE READ KEYBOARD SUPPORT                      :
                             <1>       ; INPUT                                                                  :
                             <1>       ;      (AH)= 00H  READ THE NEXT ASCII CHARACTER ENTERED FROM THE KEYBOARD,   :
                             <1>       ;               RETURN THE RESULT IN (AL), SCAN CODE IN (AH).               :
                             <1>       ;               THIS IS THE COMPATIBLE READ INTERFACE, EQUIVALENT TO THE   :
                             <1>       ;                  STANDARD PC OR PCAT KEYBOARD                                :
                             <1>       ;------------------------------------------------------------------------------:
                             <1>       ;      (AH)= 01H  SET THE ZERO FLAG TO INDICATE IF AN ASCII CHARACTER IS     :
                             <1>       ;               AVAILABLE TO BE READ FROM THE KEYBOARD BUFFER.           :
                             <1>       ;               (ZF)= 1 -- NO CODE AVAILABLE                              :
                             <1>       ;               (ZF)= 0 -- CODE IS AVAILABLE  (AX)= CHARACTER               :
                             <1>       ;               IF (ZF)= 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ IS :
                             <1>       ;               IN (AX), AND THE ENTRY REMAINS IN THE BUFFER.               :
                             <1>       ;               THIS WILL RETURN ONLY PC/PCAT KEYBOARD COMPATIBLE CODES     :
                             <1>       ;------------------------------------------------------------------------------:
                             <1>       ;      (AH)= 02H  RETURN THE CURRENT SHIFT STATUS IN AL REGISTER              :
                             <1>       ;               THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE         :
                             <1>       ;               EQUATES FOR @KB_FLAG                                         :
                             <1>       ;------------------------------------------------------------------------------:
                             <1>       ;      (AH)= 03H  SET TYPAMATIC RATE AND DELAY                                :
                             <1>       ;            (AL) = 05H                                                       :
                             <1>       ;            (BL) = TYPAMATIC RATE (BITS 5 - 7 MUST BE RESET TO 0)            :
                             <1>       ;                                                                        :
                             <1>       ;                     REGISTER     RATE      REGISTER     RATE                 :
                             <1>       ;                      VALUE     SELECTED     VALUE     SELECTED               :
                             <1>       ;                     ---------------------------------------------            :
                             <1>       ;                   00H        30.0        10H        7.5                  :
                             <1>       ;                   01H        26.7        11H        6.7                  :
                             <1>       ;                   02H        24.0        12H        6.0                  :
                             <1>       ;                   03H        21.8        13H        5.5                  :
                             <1>       ;                   04H        20.0        14H        5.0                  :
                             <1>       ;                   05H        18.5        15H        4.6                  :
                             <1>       ;                   06H        17.1        16H        4.3                  :
                             <1>       ;                   07H        16.0        17H        4.0                  :
                             <1>       ;                   08H        15.0        18H        3.7                  :
                             <1>       ;                   09H        13.3        19H        3.3                  :
                             <1>       ;                   0AH        12.0        1AH        3.0                  :
                             <1>       ;                   0BH        10.9        1BH        2.7                  :
                             <1>         ;                 0CH        10.0        1CH        2.5                  :
                             <1>       ;                   0DH         9.2        1DH        2.3                  :
                             <1>       ;                   0EH         8.6        1EH        2.1                  :
                             <1>       ;                   0FH         8.0        1FH        2.0                  :
                             <1>       ;                                                                 :
                             <1>       ;            (BH) = TYPAMATIC DELAY  (BITS 2 - 7 MUST BE RESET TO 0)          :
                             <1>       ;                                                                        :
                             <1>       ;                     REGISTER     DELAY                                       :
                             <1>       ;                      VALUE       VALUE                                       :
                             <1>       ;                     ------------------                                       :
                             <1>       ;                   00H        250 ms                                      :
                             <1>       ;                   01H        500 ms                                      :
                             <1>       ;                   02H        750 ms                                      :
                             <1>       ;                   03H       1000 ms                                      :
                             <1>       ;------------------------------------------------------------------------------:
                             <1>       ;      (AH)= 05H  PLACE ASCII CHARACTER/SCAN CODE COMBINATION IN KEYBOARD     :
                             <1>       ;               BUFFER AS IF STRUCK FROM KEYBOARD                           :
                             <1>       ;               ENTRY:  (CL) = ASCII CHARACTER                            :
                             <1>       ;                       (CH) = SCAN CODE                                    :
                             <1>       ;               EXIT:   (AH) = 00H = SUCCESSFUL OPERATION                   :
                             <1>       ;                       (AL) = 01H = UNSUCCESSFUL - BUFFER FULL             :
                             <1>       ;               FLAGS:  CARRY IF ERROR                                      :
                             <1>       ;------------------------------------------------------------------------------:
                             <1>       ;      (AH)= 10H  EXTENDED READ INTERFACE FOR THE ENHANCED KEYBOARD,          :
                             <1>       ;               OTHERWISE SAME AS FUNCTION AH=0                             :
                             <1>       ;------------------------------------------------------------------------------:
                             <1>       ;      (AH)= 11H  EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD,            :
                             <1>       ;               OTHERWISE SAME AS FUNCTION AH=1                             :
```

```
                             <1>       ;-----------------------------------------------------------------------------:
                             <1>       ;      (AH)= 12H  RETURN THE EXTENDED SHIFT STATUS IN AX REGISTER            :
                             <1>       ;               AL = BITS FROM KB_FLAG, AH = BITS FOR LEFT AND RIGHT       :
                             <1>       ;               CTL AND ALT KEYS FROM KB_FLAG_1 AND KB_FLAG_3              :
                             <1>       ; OUTPUT                                                             :
                             <1>       ;      AS NOTED ABOVE, ONLY (AX) AND FLAGS CHANGED                       :
                             <1>       ;      ALL REGISTERS RETAINED                                               :
                             <1>       ;------------------------------------------------------------------------------
                             <1>
                             <1> ; 15/01/2017
                             <1> ; 14/01/2017
                             <1> ; 02/01/2017
                             <1> ; 29/05/2016
                             <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> int32h:  ; Keyboard BIOS
                             <1>
                             <1> KEYBOARD_IO_1:
                             <1>       ;sti                      ; INTERRUPTS BACK ON
                             <1>       ; 29/05/2016
00000C1E 80642408BE          <1>         and     byte [esp+8], 10111110b ; clear zero flag and cary flag
                             <1>       ;
00000C23 1E                  <1>       push   ds                ; SAVE CURRENT DS
00000C24 53                  <1>       push   ebx               ; SAVE BX TEMPORARILY
                             <1>       ;push  ecx               ; SAVE CX TEMPORARILY
00000C25 66BB1000            <1>         mov     bx, KDATA
00000C29 8EDB                <1>       mov    ds, bx            ; PUT SEGMENT VALUE OF DATA AREA INTO DS
                             <1>
                             <1>       ; 14/01/2017
00000C2B 8B1C24              <1>       mov    ebx, [esp]
                             <1>       ;; 15/01/2017
                             <1>       ; 02/01/2017
                             <1>       ;;mov  byte [intflg], 32h  ; keyboard interrupt
00000C2E FB                  <1>       sti
                             <1>       ;
                             <1>
00000C2F 08E4                <1>       or     ah, ah            ; CHECK FOR (AH)= 00H
00000C31 743A                <1>       jz     short _K1         ; ASCII_READ
00000C33 FECC                <1>       dec    ah                    ; CHECK FOR (AH)= 01H
00000C35 7453                <1>         jz      short _K2             ; ASCII_STATUS
00000C37 FECC                <1>       dec    ah                ; CHECK FOR (AH)= 02H
00000C39 0F8494000000        <1>         jz      _K3                   ; SHIFT STATUS
00000C3F FECC                <1>       dec    ah                ; CHECK FOR (AH)= 03H
00000C41 0F8493000000        <1>         jz      _K300                 ; SET TYPAMATIC RATE/DELAY
00000C47 80EC02              <1>       sub    ah, 2             ; CHECK FOR (AH)= 05H
00000C4A 0F84BC000000        <1>         jz      _K500                 ; KEYBOARD WRITE
                             <1> _KIO1:
00000C50 80EC0B              <1>       sub    ah, 11            ; AH =  10H
00000C53 740C                <1>       jz     short _K1E        ; EXTENDED ASCII READ
00000C55 FECC                <1>       dec    ah                ; CHECK FOR (AH)= 11H
00000C57 7422                <1>       jz     short _K2E        ; EXTENDED_ASCII_STATUS
00000C59 FECC                <1>       dec    ah                ; CHECK FOR (AH)= 12H
00000C5B 7458                <1>       jz     short _K3E        ; EXTENDED_SHIFT_STATUS
                             <1> _KIO_EXIT:
                             <1>       ; 02/01/2017
00000C5D FA                  <1>       cli
                             <1>       ;;mov  byte [intflg], 0 ;; 15/01/2017
                             <1>       ;
                             <1>       ;pop   ecx               ; RECOVER REGISTER
00000C5E 5B                  <1>       pop    ebx               ; RECOVER REGISTER
00000C5F 1F                  <1>       pop    ds                ; RECOVER SEGMENT
00000C60 CF                  <1>       iretd                    ; INVALID COMMAND, EXIT
                             <1>
                             <1>       ;----- ASCII CHARACTER
                             <1> _K1E:
00000C61 E8D3000000          <1>       call   _K1S              ; GET A CHARACTER FROM THE BUFFER (EXTENDED)
00000C66 E848010000          <1>       call   _KIO_E_XLAT       ; ROUTINE TO XLATE FOR EXTENDED CALLS
00000C6B EBF0                <1>       jmp    short _KIO_EXIT       ; GIVE IT TO THE CALLER
                             <1> _K1:
00000C6D E8C7000000          <1>       call   _K1S              ; GET A CHARACTER FROM THE BUFFER
00000C72 E847010000          <1>       call   _KIO_S_XLAT       ; ROUTINE TO XLATE FOR STANDARD CALLS
00000C77 72F4                <1>       jc     short _K1         ; CARRY SET MEANS TROW CODE AWAY
                             <1> _K1A:
00000C79 EBE2                <1>       jmp    short _KIO_EXIT       ; RETURN TO CALLER
                             <1>
                             <1>       ;----- ASCII STATUS
                             <1> _K2E:
00000C7B E804010000          <1>       call   _K2S              ; TEST FOR CHARACTER IN BUFFER (EXTENDED)
00000C80 7420                <1>       jz     short _K2B        ; RETURN IF BUFFER EMPTY
00000C82 9C                  <1>       pushf                    ; SAVE ZF FROM TEST
00000C83 E82B010000          <1>       call   _KIO_E_XLAT       ; ROUTINE TO XLATE FOR EXTENDED CALLS
00000C88 EB17                <1>       jmp    short _K2A          ; GIVE IT TO THE CALLER
                             <1> _K2:
00000C8A E8F5000000          <1>       call   _K2S              ; TEST FOR CHARACTER IN BUFFER
00000C8F 7411                <1>       jz     short _K2B        ; RETURN IF BUFFER EMPTY
00000C91 9C                  <1>       pushf                    ; SAVE ZF FROM TEST
00000C92 E827010000          <1>       call   _KIO_S_XLAT       ; ROUTINE TO XLATE FOR STANDARD CALLS
00000C97 7308                <1>       jnc    short _K2A          ; CARRY CLEAR MEANS PASS VALID CODE
00000C99 9D                  <1>       popf                     ; INVALID CODE FOR THIS TYPE OF CALL
00000C9A E89A000000          <1>       call   _K1S              ; THROW THE CHARACTER AWAY
00000C9F EBE9                <1>       jmp    short _K2         ; GO LOOK FOR NEXT CHAR, IF ANY
                             <1> _K2A:
00000CA1 9D                  <1>       popf                     ; RESTORE ZF FROM TEST
                             <1> _K2B:
                             <1>       ; 02/01/2017
00000CA2 FA                  <1>       cli
                             <1>       ;; mov byte [intflg], 0 ;; 15/01/2017
                             <1>       ;
                             <1>       ;pop   ecx               ; RECOVER REGISTER
00000CA3 5B                  <1>       pop    ebx               ; RECOVER REGISTER
00000CA4 1F                  <1>       pop    ds                ; RECOVER SEGMENT
                             <1>       ; (*) 29/05/2016
                             <1>       ; (*) retf 4             ; THROW AWAY (e)FLAGS
00000CA5 7208                <1>       jc     short _k2d
```

```
00000CA7 7505                <1>         jnz    short _k2c
00000CA9 804C240840          <1>         or     byte [esp+8], 01000000b   ; set zero flag bit of eflags register
                             <1> _k2c:
00000CAE CF                  <1>         iretd
                             <1> _k2d:
                             <1>         ; 29/05/2016 -set carry flag on stack-
                             <1>         ; [esp] = EIP
                             <1>         ; [esp+4] = CS
                             <1>         ; [esp+8] = E-FLAGS
00000CAF 804C240801          <1>         or     byte [esp+8], 1  ; set carry bit of eflags register
                             <1>         ; [esp+12] = ESP (user)
                             <1>         ; [esp+16] = SS (User)
00000CB4 CF                  <1>         iretd
                             <1>
                             <1>
                             <1>         ; (*) 29/05/2016 - 'ref 4' intruction causes to stack fault
                             <1>         ; (OUTER-PRIVILEGE-LEVEL)
                             <1>         ; INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986
                             <1>         ; // RETF instruction:
                             <1>         ;
                             <1>         ; IF OperandMode=32 THEN
                             <1>         ;    Load CS:EIP from stack;
                             <1>         ;    Set CS RPL to CPL;
                             <1>         ;    Increment eSP by 8 plus the immediate offset if it exists;
                             <1>         ;    Load SS:eSP from stack;
                             <1>         ; ELSE (* OperandMode=16 *)
                             <1>         ;    Load CS:IP from stack;
                             <1>         ;    Set CS RPL to CPL;
                             <1>         ;    Increment eSP by 4 plus the immediate offset if it exists;
                             <1>         ;    Load SS:eSP from stack;
                             <1>         ; FI;
                             <1>         ;
                             <1>         ; //
                             <1>
                             <1>         ;----- SHIFT STATUS
                             <1> _K3E:                                 ; GET THE EXTENDED SHIFT STATUS FLAGS
00000CB5 8A25[8E5E0000]      <1>         mov    ah, [KB_FLAG_1]          ; GET SYSTEM SHIFT KEY STATUS
00000CBB 80E404              <1>         and    ah, SYS_SHIFT      ; MASK ALL BUT SYS KEY BIT
                             <1>         ;mov   cl, 5              ; SHIFT THEW SYSTEMKEY BIT OVER TO
                             <1>         ;shl   ah, cl             ; BIT 7 POSITION
00000CBE C0E405              <1>           shl  ah, 5
00000CC1 A0[8E5E0000]        <1>         mov    al, [KB_FLAG_1]          ; GET SYSTEM SHIFT STATES BACK
00000CC6 2473                <1>         and    al, 01110011b      ; ELIMINATE SYS SHIFT, HOLD_STATE AND INS_SHIFT
00000CC8 08C4                <1>         or     ah, al                 ; MERGE REMAINING BITS INTO AH
00000CCA A0[905E0000]        <1>         mov    al, [KB_FLAG_3]          ; GET RIGHT CTL AND ALT
00000CCF 240C                <1>         and    al, 00001100b      ; ELIMINATE LC_E0 AND LC_E1
00000CD1 08C4                <1>         or     ah, al             ; OR THE SHIFT FLAGS TOGETHER
                             <1> _K3:
00000CD3 A0[8D5E0000]        <1>         mov    al, [KB_FLAG]      ; GET THE SHIFT STATUS FLAGS
                             <1>         ;jmp   short _KIO_EXIT          ; RETURN TO CALLER
00000CD8 EB83                <1>         jmp    _KIO_EXIT
                             <1>
                             <1>         ;----- SET TYPAMATIC RATE AND DELAY
                             <1> _K300:
00000CDA 3C05                <1>         cmp    al, 5              ; CORRECT FUNCTION CALL?
                             <1>         ;jne   short _KIO_EXIT          ; NO, RETURN
00000CDC 0F857BFFFFFF        <1>         jne    _KIO_EXIT
00000CE2 F6C3E0              <1>         test   bl, 0E0h           ; TEST FOR OUT-OF-RANGE RATE
00000CE5 0F8572FFFFFF        <1>           jnz     _KIO_EXIT              ; RETURN IF SO
00000CEB F6C7FC              <1>         test   BH, 0FCh           ; TEST FOR OUT-OF-RANGE DELAY
00000CEE 0F8569FFFFFF        <1>           jnz     _KIO_EXIT              ; RETURN IF SO
00000CF4 B0F3                <1>         mov    al, KB_TYPA_RD           ; COMMAND FOR TYPAMATIC RATE/DELAY
00000CF6 E8DA060000          <1>         call   SND_DATA           ; SEND TO KEYBOARD
                             <1>         ;mov   cx, 5              ; SHIFT COUNT
                             <1>         ;shl   bh, cl             ; SHIFT DELAY OVER
00000CFB C0E705              <1>         shl    bh, 5
00000CFE 88D8                <1>         mov    al, bl             ; PUT IN RATE
00000D00 08F8                <1>         or     al, bh             ; AND DELAY
00000D02 E8CE060000          <1>         call   SND_DATA           ; SEND TO KEYBOARD
00000D07 E951FFFFFF          <1>           jmp     _KIO_EXIT              ; RETURN TO CALLER
                             <1>
                             <1>         ;----- WRITE TO KEYBOARD BUFFER
                             <1> _K500:
00000D0C 56                  <1>         push   esi                ; SAVE SI (esi)
00000D0D FA                  <1>         cli                       ;
00000D0E 8B1D[9E5E0000]      <1>         mov    ebx, [BUFFER_TAIL] ; GET THE 'IN TO' POINTER TO THE BUFFER
00000D14 89DE                <1>         mov    esi, ebx           ; SAVE A COPY IN CASE BUFFER NOT FULL
00000D16 E8D3000000          <1>         call   _K4                ; BUMP THE POINTER TO SEE IF BUFFER IS FULL
00000D1B 3B1D[9A5E0000]      <1>         cmp    ebx, [BUFFER_HEAD] ; WILL THE BUFFER OVERRUN IF WE STORE THIS?
00000D21 740D                <1>         je     short _K502        ; YES - INFORM CALLER OF ERROR
00000D23 66890E              <1>         mov    [esi], cx          ; NO - PUT ASCII/SCAN CODE INTO BUFFER
00000D26 891D[9E5E0000]      <1>         mov    [BUFFER_TAIL], ebx ; ADJUST 'IN TO' POINTER TO REFLECT CHANGE
00000D2C 28C0                <1>         sub    al, al             ; TELL CALLER THAT OPERATION WAS SUCCESSFUL
00000D2E EB02                <1>         jmp    short _K504        ; SUB INSTRUCTION ALSO RESETS CARRY FLAG
                             <1> _K502:
00000D30 B001                <1>         mov    al, 01h                  ; BUFFER FULL INDICATION
                             <1> _K504:
00000D32 FB                  <1>         sti
00000D33 5E                  <1>         pop    esi                ; RECOVER SI (esi)
00000D34 E924FFFFFF          <1>           jmp     _KIO_EXIT              ; RETURN TO CALLER WITH STATUS IN AL
                             <1>
                             <1>         ;----- READ THE KEY TO FIGURE OUT WHAT TO DO -----
                             <1> _K1S:
00000D39 FA                  <1>         cli    ; 03/12/2014
00000D3A 8B1D[9A5E0000]      <1>           mov     ebx, [BUFFER_HEAD]     ; GET POINTER TO HEAD OF BUFFER
00000D40 3B1D[9E5E0000]      <1>           cmp     ebx, [BUFFER_TAIL]     ; TEST END OF BUFFER
                             <1>         ;jne   short _K1U         ; IF ANYTHING IN BUFFER SKIP INTERRUPT
00000D46 750F                <1>         jne    short _k1x ; 03/12/2014
                             <1>         ;
                             <1>         ; 03/12/2014
                             <1>         ; 28/08/2014
                             <1>         ; PERFORM OTHER FUNCTION ?? here !
                             <1>         ;; MOV AX, 9002h           ; MOVE IN WAIT CODE & TYPE
                             <1>       ;; INT       15H                 ; PERFORM OTHER FUNCTION
                             <1> _K1T:                                   ; ASCII READ
00000D48 FB                  <1>       sti                       ; INTERRUPTS BACK ON DURING LOOP
00000D49 90                  <1>       nop                       ; ALLOW AN INTERRUPT TO OCCUR
                             <1> _K1U:
00000D4A FA                  <1>       cli                       ; INTERRUPTS BACK OFF
00000D4B 8B1D[9A5E0000]      <1>         mov        ebx, [BUFFER_HEAD]  ; GET POINTER TO HEAD OF BUFFER
00000D51 3B1D[9E5E0000]      <1>         cmp     ebx, [BUFFER_TAIL]     ; TEST END OF BUFFER
                             <1> _k1x:
00000D57 53                  <1>       push  ebx                 ; SAVE ADDRESS
00000D58 9C                  <1>       pushf                     ; SAVE FLAGS
00000D59 E82F070000          <1>       call  MAKE_LED            ; GO GET MODE INDICATOR DATA BYTE
00000D5E 8A1D[8F5E0000]      <1>       mov   bl, [KB_FLAG_2]     ; GET PREVIOUS BITS
00000D64 30C3                <1>       xor   bl, al              ; SEE IF ANY DIFFERENT
00000D66 80E307              <1>       and   bl, 07h      ; KB_LEDS    ; ISOLATE INDICATOR BITS
00000D69 7406                <1>       jz    short _K1V          ; IF NO CHANGE BYPASS UPDATE
00000D6B E8C9060000          <1>       call  SND_LED1
00000D70 FA                  <1>       cli                       ; DISABLE INTERRUPTS
                             <1> _K1V:
00000D71 9D                  <1>       popf                      ; RESTORE FLAGS
00000D72 5B                  <1>       pop   ebx                 ; RESTORE ADDRESS
00000D73 74D3                <1>         je         short _K1T           ; LOOP UNTIL SOMETHING IN BUFFER
                             <1>       ;
00000D75 668B03              <1>       mov   ax, [ebx]           ; GET SCAN CODE AND ASCII CODE
00000D78 E871000000          <1>         call    _K4                     ; MOVE POINTER TO NEXT POSITION
00000D7D 891D[9A5E0000]      <1>         mov     [BUFFER_HEAD], ebx      ; STORE VALUE IN VARIABLE
00000D83 C3                  <1>       retn                      ; RETURN
                             <1>
                             <1>       ;----- READ THE KEY TO SEE IF ONE IS PRESENT -----
                             <1> _K2S:
00000D84 FA                  <1>       cli                       ; INTERRUPTS OFF
00000D85 8B1D[9A5E0000]      <1>         mov     ebx, [BUFFER_HEAD]      ; GET HEAD POINTER
00000D8B 3B1D[9E5E0000]      <1>         cmp     ebx, [BUFFER_TAIL]      ; IF EQUAL (Z=1) THEN NOTHING THERE
00000D91 668B03              <1>       mov   ax, [ebx]
00000D94 9C                  <1>       pushf                     ; SAVE FLAGS
00000D95 6650                <1>       push  ax                  ; SAVE CODE
00000D97 E8F1060000          <1>       call  MAKE_LED            ; GO GET MODE INDICATOR DATA BYTE
00000D9C 8A1D[8F5E0000]      <1>       mov   bl, [KB_FLAG_2]     ; GET PREVIOUS BITS
00000DA2 30C3                <1>       xor   bl, al              ; SEE IF ANY DIFFERENT
00000DA4 80E307              <1>       and   bl, 07h ; KB_LEDS   ; ISOLATE INDICATOR BITS
00000DA7 7405                <1>       jz    short _K2T          ; IF NO CHANGE BYPASS UPDATE
00000DA9 E874060000          <1>       call  SND_LED                   ; GO TURN ON MODE INDICATORS
                             <1> _K2T:
00000DAE 6658                <1>       pop   ax                  ; RESTORE CODE
00000DB0 9D                  <1>       popf                      ; RESTORE FLAGS
00000DB1 FB                  <1>       sti                       ; INTERRUPTS BACK ON
00000DB2 C3                  <1>       retn                      ; RETURN
                             <1>
                             <1>       ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS -----
                             <1> _KIO_E_XLAT:
00000DB3 3CF0                <1>       cmp   al, 0F0h            ; IS IT ONE OF THE FILL-INs?
00000DB5 7506                <1>       jne   short _KIO_E_RET    ; NO, PASS IT ON
00000DB7 08E4                <1>         or  ah, ah              ; AH = 0 IS SPECIAL CASE
00000DB9 7402                <1>         jz  short _KIO_E_RET        ; PASS THIS ON UNCHANGED
00000DBB 30C0                <1>       xor   al, al              ; OTHERWISE SET AL = 0
                             <1> _KIO_E_RET:
00000DBD C3                  <1>       retn                      ; GO BACK
                             <1>
                             <1>       ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS -----
                             <1> _KIO_S_XLAT:
00000DBE 80FCE0              <1>       cmp   ah, 0E0h            ; IS IT KEYPAD ENTER OR / ?
00000DC1 750F                <1>       jne   short _KIO_S2       ; NO, CONTINUE
00000DC3 3C0D                <1>       cmp   al, 0Dh                   ; KEYPAD ENTER CODE?
00000DC5 7408                <1>         je  short _KIO_S1       ; YES, MASSAGE A BIT
00000DC7 3C0A                <1>       cmp   al, 0Ah                   ; CTRL KEYPAD ENTER CODE?
00000DC9 7404                <1>         je  short _KIO_S1       ; YES, MASSAGE THE SAME
00000DCB B435                <1>       mov   ah, 35h                   ; NO, MUST BE KEYPAD /
                             <1> _kio_ret: ; 03/12/2014
00000DCD F8                  <1>       clc
00000DCE C3                  <1>       retn
                             <1>       ;jmp  short _KIO_USE            ; GIVE TO CALLER
                             <1> _KIO_S1:
00000DCF B41C                <1>       mov   ah, 1Ch                   ; CONVERT TO COMPATIBLE OUTPUT
                             <1>       ;jmp  short _KIO_USE            ; GIVE TO CALLER
00000DD1 C3                  <1>       retn
                             <1> _KIO_S2:
00000DD2 80FC84              <1>       cmp   ah, 84h                   ; IS IT ONE OF EXTENDED ONES?
00000DD5 7715                <1>       ja    short _KIO_DIS            ; YES, THROW AWAY AND GET ANOTHER CHAR
00000DD7 3CF0                <1>       cmp   al, 0F0h            ; IS IT ONE OF THE FILL-INs?
00000DD9 7506                <1>         jne short _KIO_S3       ; NO, TRY LAST TEST
00000DDB 08E4                <1>       or    ah, ah              ; AH = 0 IS SPECIAL CASE
00000DDD 740C                <1>         jz  short _KIO_USE            ; PASS THIS ON UNCHANGED
00000DDF EB0B                <1>       jmp   short _KIO_DIS            ; THROW AWAY THE REST
                             <1> _KIO_S3:
00000DE1 3CE0                <1>       cmp   al, 0E0h            ; IS IT AN EXTENSION OF A PREVIOUS ONE?
                             <1>       ;jne  short _KIO_USE            ; NO, MUST BE A STANDARD CODE
00000DE3 75E8                <1>       jne   short _kio_ret
00000DE5 08E4                <1>       or    ah, ah              ; AH = 0 IS SPECIAL CASE
00000DE7 7402                <1>         jz  short _KIO_USE            ; JUMP IF AH = 0
00000DE9 30C0                <1>       xor   al, al              ; CONVERT TO COMPATIBLE OUTPUT
                             <1>       ;jmp  short _KIO_USE            ; PASS IT ON TO CALLER
                             <1> _KIO_USE:
                             <1>       ;clc                      ; CLEAR CARRY TO INDICATE GOOD CODE
00000DEB C3                  <1>       retn                      ; RETURN
                             <1> _KIO_DIS:
00000DEC F9                  <1>       stc                       ; SET CARRY TO INDICATE DISCARD CODE
00000DED C3                  <1>       retn                      ; RETURN
                             <1>
                             <1>       ;----- INCREMENT BUFFER POINTER ROUTINE -----
                             <1> _K4:
00000DEE 43                  <1>       inc     ebx
00000DEF 43                  <1>       inc   ebx                 ; MOVE TO NEXT WORD IN LIST
00000DF0 3B1D[965E0000]      <1>         cmp     ebx, [BUFFER_END]      ; AT END OF BUFFER?
```

```
                                <1>         ;jne    short _K5               ; NO, CONTINUE
00000DF6 7206                   <1>       jb     short _K5
00000DF8 8B1D[925E0000]         <1>         mov     ebx, [BUFFER_START]     ; YES, RESET TO BUFFER BEGINNING
                                <1> _K5:
00000DFE C3                     <1>       retn
                                <1>
                                <1> ; 20/02/2015
                                <1> ; 05/12/2014
                                <1> ; 26/08/2014
                                <1> ; KEYBOARD (HARDWARE) INTERRUPT -  IRQ LEVEL 1
                                <1> ; (INT_09h - Retro UNIX 8086 v1 - U9.ASM, 07/03/2014)
                                <1> ;
                                <1> ; Derived from "KB_INT_1" procedure of IBM "pc-at"
                                <1> ; rombios source code (06/10/1985)
                                <1> ; 'keybd.asm', HARDWARE INT 09h - (IRQ Level 1)
                                <1>
                                <1> ; EQUATES (IBM PC-XT-286 BIOS, 1986, 'POSQEQU.INC')
                                <1>
                                <1> ;--------- 8042 COMMANDS -----------------------------------------------------
                                <1> ENA_KBD           equ    0AEh         ; ENABLE KEYBOARD COMMAND
                                <1> DIS_KBD           equ    0ADh         ; DISABLE KEYBOARD COMMAND
                                <1> SHUT_CMD    equ    0FEh         ; CAUSE A SHUTDOWN COMMAND
                                <1> ;--------- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS -----------
                                <1> STATUS_PORT equ    064h         ; 8042 STATUS PORT
                                <1> INPT_BUF_FULL     equ    00000010b    ; 1 = +INPUT BUFFER FULL
                                <1> PORT_A            equ    060h         ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
                                <1> ;---------- 8042 KEYBOARD RESPONSE -------------------------------------------
                                <1> KB_ACK            equ    0FAh         ; ACKNOWLEDGE PROM TRANSMISSION
                                <1> KB_RESEND   equ    0FEh         ; RESEND REQUEST
                                <1> KB_OVER_RUN equ    0FFh         ; OVER RUN SCAN CODE
                                <1> ;---------- KEYBOARD/LED COMMANDS --------------------------------------------
                                <1> KB_ENABLE   equ    0F4h         ; KEYBOARD ENABLE
                                <1> LED_CMD           equ    0EDh         ; LED WRITE COMMAND
                                <1> KB_TYPA_RD  equ    0F3h         ; TYPAMATIC RATE/DELAY COMMAND
                                <1> ;---------- KEYBOARD SCAN CODES ------------------------------------------------
                                <1> NUM_KEY           equ    69           ; SCAN CODE FOR      NUMBER LOCK KEY
                                <1> SCROLL_KEY  equ    70           ; SCAN CODE FOR      SCROLL LOCK KEY
                                <1> ALT_KEY           equ    56           ; SCAN CODE FOR      ALTERNATE SHIFT KEY
                                <1> CTL_KEY           equ    29           ; SCAN CODE FOR      CONTROL KEY
                                <1> CAPS_KEY    equ    58           ; SCAN CODE FOR      SHIFT LOCK KEY
                                <1> DEL_KEY           equ    83           ; SCAN CODE FOR      DELETE KEY
                                <1> INS_KEY           equ    82           ; SCAN CODE FOR      INSERT KEY
                                <1> LEFT_KEY    equ    42           ; SCAN CODE FOR      LEFT SHIFT
                                <1> RIGHT_KEY   equ    54           ; SCAN CODE FOR      RIGHT SHIFT
                                <1> SYS_KEY           equ    84           ; SCAN CODE FOR      SYSTEM KEY
                                <1> ;---------- ENHANCED KEYBOARD SCAN CODES --------------------------------------
                                <1> ID_1        equ    0ABh         ; 1ST ID CHARACTER FOR KBX
                                <1> ID_2        equ    041h         ; 2ND ID CHARACTER FOR KBX
                                <1> ID_2A       equ    054h         ; ALTERNATE 2ND ID CHARACTER FOR KBX
                                <1> F11_M       equ    87           ; F11 KEY MAKE
                                <1> F12_M       equ    88           ; F12 KEY MAKE
                                <1> MC_E0       equ    224          ; GENERAL MARKER CODE
                                <1> MC_E1       equ    225          ; PAUSE KEY MARKER CODE
                                <1> ;---------- FLAG EQUATES WITHIN @KB_FLAG-----------------------------------------
                                <1> RIGHT_SHIFT equ    00000001b    ; RIGHT SHIFT KEY DEPRESSED
                                <1> LEFT_SHIFT  equ    00000010b    ; LEFT SHIFT KEY DEPRESSED
                                <1> CTL_SHIFT   equ    00000100b    ; CONTROL SHIFT KEY DEPRESSED
                                <1> ALT_SHIFT   equ    00001000b    ; ALTERNATE SHIFT KEY DEPRESSED
                                <1> SCROLL_STATE equ   00010000b    ; SCROLL LOCK STATE IS ACTIVE
                                <1> NUM_STATE   equ    00100000b    ; NUM LOCK STATE IS ACTIVE
                                <1> CAPS_STATE  equ    01000000b    ; CAPS LOCK STATE IS ACTIVE
                                <1> INS_STATE   equ    10000000b    ; INSERT STATE IS ACTIVE
                                <1> ;---------- FLAG EQUATES WITHIN @KB_FLAG_1 ------------------------------------
                                <1> L_CTL_SHIFT equ    00000001b    ; LEFT CTL KEY DOWN
                                <1> L_ALT_SHIFT equ    00000010b    ; LEFT ALT KEY DOWN
                                <1> SYS_SHIFT   equ    00000100b    ; SYSTEM KEY DEPRESSED AND HELD
                                <1> HOLD_STATE  equ    00001000b    ; SUSPEND KEY HAS BEEN TOGGLED
                                <1> SCROLL_SHIFT equ   00010000b    ; SCROLL LOCK KEY IS DEPRESSED
                                <1> NUM_SHIFT   equ    00100000b    ; NUM LOCK KEY IS DEPRESSED
                                <1> CAPS_SHIFT  equ    01000000b    ; CAPS LOCK KEY IS DEPRE55ED
                                <1> INS_SHIFT   equ    10000000b    ; INSERT KEY IS DEPRESSED
                                <1> ;---------- FLAGS EQUATES WITHIN @KB_FLAG_2 ----------------------------------
                                <1> KB_LEDS           equ    00000111b    ; KEYBOARD LED STATE BITS
                                <1> ;           equ    00000001b    ; SCROLL LOCK INDICATOR
                                <1> ;           equ    00000010b    ; NUM LOCK INDICATOR
                                <1> ;           equ    00000100b    ; CAPS LOCK INDICATOR
                                <1> ;           equ    00001000b    ; RESERVED (MUST BE ZERO)
                                <1> KB_FA       equ    00010000b    ; ACKNOWLEDGMENT RECEIVED
                                <1> KB_FE       equ    00100000b    ; RESEND RECEIVED FLAG
                                <1> KB_PR_LED   equ    01000000b    ; MODE INDICATOR UPDATE
                                <1> KB_ERR            equ    10000000b    ; KEYBOARD TRANSMIT ERROR FLAG
                                <1> ;----------- FLAGS EQUATES WITHIN @KB_FLAG_3 ----------------------------------
                                <1> LC_E1       equ    00000001b    ; LAST CODE WAS THE E1 HIDDEN CODE
                                <1> LC_E0       equ    00000010b    ; LAST CODE WAS THE E0 HIDDEN CODE
                                <1> R_CTL_SHIFT equ    00000100b    ; RIGHT CTL KEY DOWN
                                <1> R_ALT_SHIFT equ    00001000b    ; RIGHT ALT KEY DOWN
                                <1> GRAPH_ON    equ    00001000b    ; ALT GRAPHICS KEY DOWN (WT ONLY)
                                <1> KBX         equ    00010000b    ; ENHANCED KEYBOARD INSTALLED
                                <1> SET_NUM_LK  equ    00100000b    ; FORCE NUM LOCK IF READ ID AND KBX
                                <1> LC_AB       equ    01000000b    ; LAST CHARACTER WAS FIRST ID CHARACTER
                                <1> RD_ID       equ    10000000b    ; DOING A READ ID (MUST BE BIT0)
                                <1> ;
                                <1> ;----------- INTERRUPT EQUATES ------------------------------------------------
                                <1> EOI         equ    020h         ; END OF INTERRUPT COMMAND TO 8259
                                <1> INTA00            equ    020h         ; 8259 PORT
                                <1>
                                <1>
                                <1> kb_int:
                                <1>
                                <1> ; 17/10/2015 ('ctrlbrk')
                                <1> ; 05/12/2014
                                <1> ; 04/12/2014 (derived from pc-xt-286 bios source code -1986-)
                                <1> ; 26/08/2014
```

```
                             <1> ;
                             <1> ; 03/06/86  KEYBOARD BIOS
                             <1> ;
                             <1> ;--- HARDWARE INT 09H -- (IRQ LEVEL 1) -----------------------------------------
                             <1> ;                                                                ;
                             <1> ;     KEYBOARD INTERRUPT ROUTINE                                 ;
                             <1> ;                                                                ;
                             <1> ;--------------------------------------------------------------------------------
                             <1> 
                             <1> KB_INT_1:
00000DFF FB                  <1>       sti                       ; ENABLE INTERRUPTS
                             <1>       ;push ebp
00000E00 50                  <1>       push  eax
00000E01 53                  <1>       push  ebx
00000E02 51                  <1>       push  ecx
00000E03 52                  <1>       push  edx
00000E04 56                  <1>       push  esi
00000E05 57                  <1>       push  edi
00000E06 1E                  <1>       push  ds
00000E07 06                  <1>       push  es
00000E08 FC                  <1>       cld                       ; FORWARD DIRECTION
00000E09 66B81000            <1>       mov   ax, KDATA
00000E0D 8ED8                <1>       mov   ds, ax
00000E0F 8EC0                <1>       mov   es, ax
                             <1>       ;
                             <1>       ;----- WAIT FOR KEYBOARD DISABLE COMMAND TO BE ACCEPTED
00000E11 B0AD                <1>       mov   al, DIS_KBD         ; DISABLE THE KEYBOARD COMMAND
00000E13 E8A9050000          <1>       call  SHIP_IT                   ; EXECUTE DISABLE
00000E18 FA                  <1>       cli                       ; DISABLE INTERRUPTS
00000E19 B900000100          <1>       mov   ecx, 10000h         ; SET MAXIMUM TIMEOUT
                             <1> KB_INT_01:
00000E1E E464                <1>       in    al, STATUS_PORT           ; READ ADAPTER STATUS
00000E20 A802                <1>       test  al, INPT_BUF_FULL   ; CHECK INPUT BUFFER FULL STATUS BIT
00000E22 E0FA                <1>       loopnz KB_INT_01          ; WAIT FOR COMMAND TO BE ACCEPTED
                             <1>       ;
                             <1>       ;----- READ CHARACTER FROM KEYBOARD INTERFACE
00000E24 E460                <1>       in    al, PORT_A          ; READ IN THE CHARACTER
                             <1>       ;
                             <1>       ;----- SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INT LEVEL 9H)
                             <1>       ;MOV  AH, 04FH            ; SYSTEM INTERCEPT - KEY CODE FUNCTION
                             <1>       ;STC                      ; SET CY=1 (IN CASE OF IRET)
                             <1>       ;INT  15H                 ; CASETTE CALL (AL)=KEY SCAN CODE
                             <1>       ;                         ; RETURNS CY=1 FOR INVALID FUNCTION
                             <1>       ;JC   KB_INT_02           ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
                             <1>       ;JMP  K26                 ; EXIT IF SYSTEM HANDLES SCAN CODE
                             <1>       ;                         ; EXİT HANDLES HARDWARE EOI AND ENABLE
                             <1>       ;
                             <1>       ;----- CHECK FOR A RESEND COMMAND TO KEYBOARD
                             <1> KB_INT_02:                      ;       (AL)= SCAN CODE
00000E26 FB                  <1>       sti                       ; ENABLE INTERRUPTS AGAIN
00000E27 3CFE                <1>       cmp   al, KB_RESEND       ; IS THE INPUT A RESEND
00000E29 7411                <1>         je      short KB_INT_4          ; GO IF RESEND
                             <1>       ;
                             <1>       ;----- CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD
00000E2B 3CFA                <1>       cmp   al, KB_ACK          ; IS THE INPUT AN ACKNOWLEDGE
00000E2D 751A                <1>         jne     short KB_INT_2          ; GO IF NOT
                             <1>       ;
                             <1>       ;----- A COMMAND TO THE KEYBOARD WAS ISSUED
00000E2F FA                  <1>       cli                       ; DISABLE INTERRUPTS
00000E30 800D[8F5E0000]10    <1>       or    byte [KB_FLAG_2], KB_FA ; INDICATE ACK RECEIVED
00000E37 E97A020000          <1>         jmp     K26                     ; RETURN IF NOT (ACK RETURNED FOR DATA)
                             <1>       ;
                             <1>       ;----- RESEND THE LAST BYTE
                             <1> KB_INT_4:
00000E3C FA                  <1>       cli                       ; DISABLE INTERRUPTS
00000E3D 800D[8F5E0000]20    <1>       or    byte [KB_FLAG_2], KB_FE ; INDICATE RESEND RECEIVED
00000E44 E96D020000          <1>         jmp     K26                     ; RETURN IF NOT ACK RETURNED FOR DATA)
                             <1>       ;
                             <1> ;-----      UPDATE MODE INDICATORS IF CHANGE IN STATE
                             <1> KB_INT_2:
00000E49 6650                <1>       push  ax                  ; SAVE DATA IN
00000E4B E83D060000          <1>       call  MAKE_LED            ; GO GET MODE INDICATOR DATA BYTE
00000E50 8A1D[8F5E0000]      <1>       mov   bl, [KB_FLAG_2]     ; GET PREVIOUS BITS
00000E56 30C3                <1>       xor   bl, al              ; SEE IF ANY DIFFERENT
00000E58 80E307              <1>       and   bl, KB_LEDS         ; ISOLATE INDICATOR BITS
00000E5B 7405                <1>       jz    short UP0           ; IF NO CHANGE BYPASS UPDATE
00000E5D E8C0050000          <1>       call  SND_LED                   ; GO TURN ON MODE INDICATORS
                             <1> UP0:
00000E62 6658                <1>       pop   ax                  ; RESTORE DATA IN
                             <1> ;------------------------------------------------------------------------
                             <1> ;     START OF KEY PROCESSING                                    ;
                             <1> ;------------------------------------------------------------------------
00000E64 88C4                <1>       mov   ah, al              ; SAVE SCAN CODE IN AH ALSO
                             <1>       ;
                             <1>       ;----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
00000E66 3CFF                <1>       cmp   al, KB_OVER_RUN           ; IS THIS AN OVERRUN CHAR
00000E68 0F843F050000        <1>         je      K62                   ; BUFFER_FULL_BEEP
                             <1>       ;
                             <1> K16:
00000E6E 8A3D[905E0000]      <1>       mov   bh, [KB_FLAG_3]           ; LOAD FLAGS FOR TESTING
                             <1>       ;
                             <1>       ;----- TEST TO SEE IF A READ_ID IS IN PROGRESS
00000E74 F6C7C0              <1>       test  bh, RD_ID+LC_AB     ; ARE WE DOING A READ ID?
00000E77 7449                <1>       jz    short NOT_ID        ; CONTINUE IF NOT
00000E79 7917                <1>       jns   short TST_ID_2            ; IS THE RD_ID FLAG ON?
00000E7B 3CAB                <1>       cmp   al, ID_1            ; IS THIS THE 1ST ID CHARACTER?
00000E7D 7507                <1>       jne   short RST_RD_ID
00000E7F 800D[905E0000]40    <1>       or    byte [KB_FLAG_3], LC_AB ; INDICATE 1ST ID WAS OK
                             <1> RST_RD_ID:
00000E86 8025[905E0000]7F    <1>       and   byte [KB_FLAG_3], ~RD_ID ; RESET THE READ ID FLAG
                             <1>         ;jmp    short ID_EX           ; AND EXIT
00000E8D E924020000          <1>       jmp   K26
                             <1>       ;
```

```
                              <1> TST_ID_2:
00000E92 8025[905E0000]BF     <1>       and   byte [KB_FLAG_3], ~LC_AB ; RESET FLAG
00000E99 3C54                 <1>       cmp   al, ID_2A           ; IS THIS THE 2ND ID CHARACTER?
00000E9B 7419                 <1>         je  short KX_BIT        ; JUMP IF SO
00000E9D 3C41                 <1>       cmp   al, ID_2            ; IS THIS THE 2ND ID CHARACTER?
                              <1>         ;jne short ID_EX        ; LEAVE IF NOT
00000E9F 0F8511020000         <1>       jne   K26
                              <1>       ;
                              <1>       ;----- A READ ID SAID THAT IT WAS ENHANCED KEYBOARD
00000EA5 F6C720               <1>       test  bh, SET_NUM_LK            ; SHOULD WE SET NUM LOCK?
00000EA8 740C                 <1>         jz      short KX_BIT          ; EXIT IF NOT
00000EAA 800D[8D5E0000]20     <1>       or    byte [KB_FLAG], NUM_STATE ; FORCE NUM LOCK ON
00000EB1 E86C050000           <1>       call  SND_LED                   ; GO SET THE NUM LOCK INDICATOR
                              <1> KX_BIT:
00000EB6 800D[905E0000]10     <1>       or    byte [KB_FLAG_3], KBX     ; INDICATE ENHANCED KEYBOARD WAS FOUND
00000EBD E9F4010000           <1> ID_EX:      jmp     K26               ; EXIT
                              <1>       ;
                              <1> NOT_ID:
00000EC2 3CE0                 <1>       cmp   al, MC_E0           ; IS THIS THE GENERAL MARKER CODE?
00000EC4 750C                 <1>       jne   short TEST_E1
00000EC6 800D[905E0000]12     <1>       or    byte [KB_FLAG_3], LC_E0+KBX ; SET FLAG BIT, SET KBX, AND
                              <1>       ;jmp  short EXIT          ; THROW AWAY THIS CODE
00000ECD E9EB010000           <1>       jmp   K26A
                              <1> TEST_E1:
00000ED2 3CE1                 <1>       cmp   al, MC_E1           ; IS THIS THE PAUSE KEY?
00000ED4 750C                 <1>       jne   short NOT_HC
00000ED6 800D[905E0000]11     <1>       or    byte [KB_FLAG_3], LC_E1+KBX ; SET FLAG BIT, SET KBX, AND
00000EDD E9DB010000           <1> EXIT: jmp   K26A                ; THROW AWAY THIS CODE
                              <1>       ;
                              <1> NOT_HC:
00000EE2 247F                 <1>       and   al, 07Fh            ; TURN OFF THE BREAK BIT
00000EE4 F6C702               <1>       test  bh, LC_E0           ; LAST CODE THE E0 MARKER CODE
00000EE7 7414                 <1>       jz    short NOT_LC_E0           ; JUMP IF NOT
                              <1>       ;
00000EE9 BF[7A5D0000]         <1>       mov   edi, _K6+6          ; IS THIS A SHIFT KEY?
00000EEE AE                   <1>       scasb
00000EEF 0F84C1010000         <1>         je      K26 ; K16B             ; YES, THROW AWAY & RESET FLAG
00000EF5 AE                   <1>       scasb
00000EF6 757C                 <1>       jne   short K16A          ; NO, CONTINUE KEY PROCESSING
                              <1>       ;jmp  short K16B          ; YES, THROW AWAY & RESET FLAG
00000EF8 E9B9010000           <1>       jmp   K26
                              <1>       ;
                              <1> NOT_LC_E0:
00000EFD F6C701               <1>       test  bh, LC_E1           ; LAST CODE THE E1 MARKER CODE?
00000F00 7435                 <1>       jz    short T_SYS_KEY           ; JUMP IF NOT
00000F02 B904000000           <1>       mov   ecx, 4              ; LENGHT OF SEARCH
00000F07 BF[785D0000]         <1>       mov   edi, _K6+4          ; IS THIS AN ALT, CTL, OR SHIFT?
00000F0C F2AE                 <1>       repne scasb               ; CHECK IT
                              <1>       ;je   short EXIT          ; THROW AWAY IF SO
00000F0E 0F84A9010000         <1>       je    K26A
                              <1>       ;
00000F14 3C45                 <1>       cmp   al, NUM_KEY         ; IS IT THE PAUSE KEY?
                              <1>       ;jne  short K16B          ; NO, THROW AWAY & RESET FLAG
00000F16 0F859A010000         <1>       jne   K26
00000F1C F6C480               <1>       test  ah, 80h                   ; YES, IS IT THE BREAK OF THE KEY?
                              <1>       ;jnz  short K16B          ; YES, THROW THIS AWAY, TOO
00000F1F 0F8591010000         <1>       jnz   K26
                              <1>         ; 20/02/2015
00000F25 F605[8E5E0000]08     <1>       test  byte [KB_FLAG_1],HOLD_STATE ;  NO, ARE WE PAUSED ALREADY?
                              <1>       ;jnz  short K16B          ;  YES, THROW AWAY
00000F2C 0F8584010000         <1>       jnz   K26
00000F32 E9E1020000           <1>       jmp     K39P                   ; NO, THIS IS THE REAL PAUSE STATE
                              <1>       ;
                              <1>       ;----- TEST FOR SYSTEM KEY
                              <1> T_SYS_KEY:
00000F37 3C54                 <1>       cmp   al, SYS_KEY         ; IS IT THE SYSTEM KEY?
00000F39 7539                 <1>       jnz   short K16A          ; CONTINUE IF NOT
                              <1>       ;
00000F3B F6C480               <1>       test  ah, 80h                   ; CHECK IF THIS A BREAK CODE
00000F3E 7524                 <1>       jnz   short K16C          ; DO NOT TOUCH SYSTEM INDICATOR IF TRUE
                              <1>       ;
00000F40 F605[8E5E0000]04     <1>       test  byte [KB_FLAG_1], SYS_SHIFT ; SEE IF IN SYSTEM KEY HELD DOWN
                              <1>         ;jnz short K16B         ; IF YES, DO NOT PROCESS SYSTEM INDICATOR
00000F47 0F8569010000         <1>       jnz     K26
                              <1>       ;
00000F4D 800D[8E5E0000]04     <1>       or    byte [KB_FLAG_1], SYS_SHIFT ; INDICATE SYSTEM KEY DEPRESSED
00000F54 B020                 <1>       mov   al, EOI                   ; END OF INTERRUPT COMMAND
00000F56 E620                 <1>       out   20h, al ;out INTA00, al   ; SEND COMMAND TO INTERRUPT CONTROL PORT
                              <1>                                 ; INTERRUPT-RETURN-NO-EOI
00000F58 B0AE                 <1>       mov   al, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
00000F5A E862040000           <1>       call  SHIP_IT                   ; EXECUTE ENABLE
                              <1>       ; !!! SYSREQ !!! function/system call (INTERRUPT) must be here !!!
                              <1>       ;MOV  AL, 8500H           ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
                              <1>       ;STI                      ; MAKE SURE INTERRUPTS ENABLED
                              <1>       ;INT  15H                 ; USER INTERRUPT
00000F5F E965010000           <1>         jmp     K27A                   ; END PROCESSING
                              <1>       ;
                              <1> ;K16B:      jmp    K26                ; IGNORE SYSTEM KEY
                              <1>       ;
                              <1> K16C:
00000F64 8025[8E5E0000]FB     <1>       and   byte [KB_FLAG_1], ~SYS_SHIFT ; TURN OFF SHIFT KEY HELD DOWN
00000F6B B020                 <1>       mov   al, EOI                   ; END OF INTERRUPT COMMAND
00000F6D E620                 <1>       out   20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
                              <1>                                 ; INTERRUPT-RETURN-NO-EOI
                              <1>       ;MOV  AL, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
                              <1>       ;CALL SHIP_IT                   ; EXECUTE ENABLE
                              <1>       ;
                              <1>       ;MOV  AX, 8501H           ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY
                              <1>       ;STI                      ; MAKE SURE INTERRUPTS ENABLED
                              <1>       ;INT  15H                 ; USER INTERRUPT
                              <1>       ;JMP  K27A                ; INGONRE SYSTEM KEY
                              <1>       ;
00000F6F E94E010000           <1>       jmp     K27               ; IGNORE SYSTEM KEY
```

```
                                  <1>         ;
                                  <1>         ;----- TEST FOR SHIFT KEYS
                                  <1> K16A:
00000F74 8A1D[8D5E0000]           <1>         mov   bl, [KB_FLAG]       ; PUT STATE FLAGS IN BL
00000F7A BF[745D0000]             <1>         mov   edi, _K6            ; SHIFT KEY TABLE offset
00000F7F B908000000               <1>         mov   ecx, _K6L           ; LENGTH
00000F84 F2AE                     <1>         repne scasb               ; LOOK THROUGH THE TABLE FOR A MATCH
00000F86 88E0                     <1>         mov   al, ah              ; RECOVER SCAN CODE
00000F88 0F8510010000             <1>           jne     K25                     ; IF NO MATCH, THEN SHIFT NOT FOUND
                                  <1>         ;
                                  <1>         ;------      SHIFT KEY FOUND
                                  <1> K17:
00000F8E 81EF[755D0000]           <1>           sub     edi, _K6+1              ; ADJUST PTR TO SCAN CODE MATCH
00000F94 8AA7[7C5D0000]           <1>                 mov     ah, [edi+_K7]           ; GET MASK INTO AH
00000F9A B102                     <1>         mov   cl, 2               ; SETUP COUNT FOR FLAG SHIFTS
00000F9C A880                     <1>         test  al, 80h                     ; TEST FOR BREAK KEY
00000F9E 0F8596000000             <1>           jnz     K23                     ; JUMP OF BREAK
                                  <1>         ;
                                  <1>         ;----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
                                  <1> K17C:
00000FA4 80FC10                   <1>         cmp   ah, SCROLL_SHIFT
00000FA7 732B                     <1>         jae   short K18           ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
                                  <1>         ;
                                  <1>         ;----- PLAIN SHIFT KEY, SET SHIFT ON
00000FA9 0825[8D5E0000]           <1>         or    [KB_FLAG], ah       ; TURN ON SHIFT BIT
00000FAF A80C                     <1>           test al, CTL_SHIFT+ALT_SHIFT ; IS IT ALT OR CTRL?
                                  <1>         ;jnz  short K17D          ; YES, MORE FLAGS TO SET
00000FB1 0F84FF000000             <1>         jz    K26                 ; NO, INTERRUPT RETURN
                                  <1> K17D:
00000FB7 F6C702                   <1>         test  bh, LC_E0           ; IS THIS ONE OF NEW KEYS?
00000FBA 740B                     <1>         jz    short K17E          ; NO, JUMP
00000FBC 0825[905E0000]           <1>         or    [KB_FLAG_3], ah             ; SET BITS FOR RIGHT CTRL, ALT
00000FC2 E9EF000000               <1>         jmp   K26                 ; INTERRUPT RETURN
                                  <1> K17E:
00000FC7 D2EC                     <1>         shr   ah, cl              ; MOVE FLAG BITS TWO POSITIONS
00000FC9 0825[8E5E0000]           <1>         or    [KB_FLAG_1], ah             ; SET BITS FOR LEFT CTRL, ALT
00000FCF E9E2000000               <1>         jmp   K26
                                  <1>         ;
                                  <1>         ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
                                  <1> K18:                            ; SHIFT-TOGGLE
00000FD4 F6C304                   <1>         test  bl, CTL_SHIFT               ; CHECK CTL SHIFT STATE
                                  <1>           ;jz        short K18A             ; JUMP IF NOT CTL STATE
00000FD7 0F85C1000000             <1>           jnz     K25                     ; JUMP IF CTL STATE
                                  <1> K18A:
00000FDD 3C52                     <1>         cmp   al, INS_KEY         ; CHECK FOR INSERT KEY
00000FDF 7524                     <1>         jne   short K22           ; JUMP IF NOT INSERT KEY
00000FE1 F6C308                   <1>         test  bl, ALT_SHIFT               ; CHECK FOR ALTERNATE SHIFT
                                  <1>               ;jz    short K18B          ; JUMP IF NOT ALTERNATE SHIFT
00000FE4 0F85B4000000             <1>           jnz     K25                     ; JUMP IF ALTERNATE SHIFT
                                  <1> K18B:
00000FEA F6C702                   <1>         test  bh, LC_E0 ;20/02/2015       ; IS THIS NEW INSERT KEY?
00000FED 7516                     <1>         jnz   short K22           ; YES, THIS ONE'S NEVER A '0'
                                  <1> K19:
00000FEF F6C320                   <1>         test  bl, NUM_STATE               ; CHECK FOR BASE STATE
00000FF2 750C                     <1>         jnz   short K21           ; JUMP IF NUM LOCK IS ON
00000FF4 F6C303                   <1>         test  bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
00000FF7 740C                     <1>         jz    short K22           ; JUMP IF BASE STATE
                                  <1> K20:                            ; NUMERIC ZERO, NOT INSERT KEY
00000FF9 88C4                     <1>         mov   ah, al              ; PUT SCAN CODE BACK IN AH
00000FFB E99E000000               <1>           jmp     K25                     ; NUMERAL '0', STNDRD. PROCESSING
                                  <1> K21:                            ; MIGHT BE NUMERIC
00001000 F6C303                   <1>         test  bl, LEFT_SHIFT+RIGHT_SHIFT
00001003 74F4                     <1>         jz    short K20           ; IS NUMERIC, STD. PROC.
                                  <1>         ;
                                  <1> K22:                            ; SHIFT TOGGLE KEY HIT; PROCESS IT
00001005 8425[8E5E0000]           <1>         test  ah, [KB_FLAG_1]     ; IS KEY ALREADY DEPRESSED
0000100B 0F85A5000000             <1>           jnz     K26                     ; JUMP IF KEY ALREADY DEPRESSED
                                  <1> K22A:
00001011 0825[8E5E0000]           <1>           or      [KB_FLAG_1], ah ; INDICATE THAT THE KEY IS DEPRESSED
00001017 3025[8D5E0000]           <1>         xor   [KB_FLAG], ah       ; TOGGLE THE SHIFT STATE
                                  <1>         ;
                                  <1>         ;----- TOGGLE LED IF CAPS, NUM  OR SCROLL KEY DEPRESSED
0000101D F6C470                   <1>         test  ah, CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?
00001020 7409                     <1>         jz    short K22B          ; GO IF NOT
                                  <1>         ;
00001022 6650                     <1>         push  ax                  ; SAVE SCAN CODE AND SHIFT MASK
00001024 E8F9030000               <1>         call  SND_LED                     ; GO TURN MODE INDICATORS ON
00001029 6658                     <1>         pop   ax                  ; RESTORE SCAN CODE
                                  <1> K22B:
0000102B 3C52                     <1>         cmp   al, INS_KEY         ; TEST FOR 1ST MAKE OF INSERT KEY
0000102D 0F8583000000             <1>           jne     K26                     ; JUMP IF NOT INSERT KEY
00001033 88C4                     <1>         mov   ah, al                ; SCAN CODE IN BOTH HALVES OF AX
00001035 E999000000               <1>           jmp     K28                     ; FLAGS UPDATED, PROC. FOR BUFFER
                                  <1>         ;
                                  <1>         ;----- BREAK SHIFT FOUND
                                  <1> K23:                            ; BREAK-SHIFT-FOUND
0000103A 80FC10                   <1>         cmp   ah, SCROLL_SHIFT    ; IS THIS A TOGGLE KEY
0000103D F6D4                     <1>         not   ah                  ; INVERT MASK
0000103F 7355                     <1>         jae   short K24           ; YES, HANDLE BREAK TOGGLE
00001041 2025[8D5E0000]           <1>         and   [KB_FLAG], ah       ; TURN OFF SHIFT BIT
00001047 80FCFB                   <1>         cmp   ah, ~CTL_SHIFT              ; IS THIS ALT OR CTL?
0000104A 7730                     <1>         ja    short K23D          ; NO, ALL DONE
                                  <1>         ;
0000104C F6C702                   <1>         test  bh, LC_E0           ; 2ND ALT OR CTL?
0000104F 7408                     <1>         jz    short K23A          ; NO, HANSLE NORMALLY
00001051 2025[905E0000]           <1>         and   [KB_FLAG_3], ah             ; RESET BIT FOR RIGHT ALT OR CTL
00001057 EB08                     <1>         jmp   short K23B          ; CONTINUE
                                  <1> K23A:
00001059 D2FC                     <1>         sar   ah, cl              ; MOVE THE MASK BIT TWO POSITIONS
0000105B 2025[8E5E0000]           <1>         and   [KB_FLAG_1], ah             ; RESET BIT FOR LEFT ALT AND CTL
                                  <1> K23B:
00001061 88C4                     <1>         mov   ah, al              ; SAVE SCAN CODE
00001063 A0[905E0000]             <1>         mov   al, [KB_FLAG_3]             ; GET RIGHT ALT & CTRL FLAGS
```

```
00001068 D2E8                <1>       shr   al, cl              ; MOVE TO BITS 1 & 0
0000106A 0A05[8E5E0000]      <1>       or    al, [KB_FLAG_1]           ; PUT IN LEFT ALŞT & CTL FLAGS
00001070 D2E0                <1>       shl   al, cl              ; MOVE BACK TO BITS 3 & 2
00001072 240C                <1>       and   al, ALT_SHIFT+CTL_SHIFT ; FILTER OUT OTHER GARBAGE
00001074 0805[8D5E0000]      <1>       or    [KB_FLAG], al       ; PUT RESULT IN THE REAL FLAGS
0000107A 88E0                <1>       mov   al, ah
                             <1> K23D:
0000107C 3CB8                <1>       cmp   al, ALT_KEY+80h           ; IS THIS ALTERNATE SHIFT RELEASE
0000107E 7536                <1>       jne   short K26           ; INTERRUPT RETURN
                             <1>       ;
                             <1>       ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
00001080 A0[915E0000]        <1>       mov   al, [ALT_INPUT]
00001085 B400                <1>       mov   ah, 0               ; SCAN CODE OF 0
00001087 8825[915E0000]      <1>       mov   [ALT_INPUT], ah     ; ZERO OUT THE FIELD
0000108D 3C00                <1>       cmp   al, 0               ; WAS THE INPUT = 0?
0000108F 7425                <1>       je    short K26           ; INTERRUPT_RETURN
                             <1>         ; 29/01/2016
                             <1>       ;jmp     K61                     ; IT WASN'T, SO PUT IN BUFFER
00001091 E9D0020000          <1>       jmp   _K60
                             <1>       ;
                             <1> K24:                            ; BREAK-TOGGLE
00001096 2025[8E5E0000]      <1>       and   [KB_FLAG_1], ah     ; INDICATE NO LONGER DEPRESSED
0000109C EB18                <1>       jmp   short K26           ; INTERRUPT_RETURN
                             <1>       ;
                             <1>       ;----- TEST FOR HOLD STATE
                             <1>                                 ; AL, AH = SCAN CODE
                             <1> K25:                            ; NO-SHIFT-FOUND
0000109E 3C80                <1>       cmp   al, 80h                   ; TEST FOR BREAK KEY
000010A0 7314                <1>       jae   short K26           ; NOTHING FOR BREAK CHARS FROM HERE ON
000010A2 F605[8E5E0000]08    <1>       test  byte [KB_FLAG_1], HOLD_STATE ; ARE WE IN HOLD STATE
000010A9 7428                <1>       jz    short K28           ; BRANCH AROUND TEST IF NOT
000010AB 3C45                <1>       cmp   al, NUM_KEY
000010AD 7407                <1>       je    short K26           ; CAN'T END HOLD ON NUM_LOCK
000010AF 8025[8E5E0000]F7    <1>       and   byte [KB_FLAG_1], ~HOLD_STATE ; TURN OFF THE HOLD STATE BIT
                             <1>       ;
                             <1> K26:
000010B6 8025[905E0000]FC    <1>       and   byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
                             <1> K26A:                           ; INTERRUPT-RETURN
000010BD FA                  <1>       cli                       ; TURN OFF INTERRUPTS
000010BE B020                <1>       mov   al, EOI                   ; END OF INTERRUPT COMMAND
000010C0 E620                <1>       out   20h, al      ;out INTA00, al     ; SEND COMMAND TO INTERRUPT CONTROL PORT
                             <1> K27:                            ; INTERRUPT-RETURN-NO-EOI
000010C2 B0AE                <1>       mov   al, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
000010C4 E8F8020000          <1>       call  SHIP_IT                   ; EXECUTE ENABLE
                             <1> K27A:
000010C9 FA                  <1>       cli                       ; DISABLE INTERRUPTS
                             <1>       ;;mov byte [intflg], 0 ; 07/01/2017 ;; 15/01/2017
000010CA 07                  <1>       pop   es                  ; RESTORE REGISTERS
000010CB 1F                  <1>       pop   ds
000010CC 5F                  <1>       pop   edi
000010CD 5E                  <1>       pop   esi
000010CE 5A                  <1>       pop   edx
000010CF 59                  <1>       pop   ecx
000010D0 5B                  <1>       pop   ebx
000010D1 58                  <1>       pop   eax
                             <1>       ;pop  ebp
000010D2 CF                  <1>       iretd                     ; RETURN
                             <1>
                             <1>       ;----- NOT IN HOLD STATE
                             <1> K28:                            ; NO-HOLD-STATE
000010D3 3C58                <1>       cmp   al, 88              ; TEST FOR OUT-OF-RANGE SCAN CODES
000010D5 77DF                <1>       ja    short K26           ; IGNORE IF OUT-OF-RANGE
                             <1>       ;
000010D7 F6C308              <1>       test  bl, ALT_SHIFT             ; ARE WE IN ALTERNATE SHIFT
                             <1>         ;jz short K28A          ; IF NOT ALTERNATE
000010DA 0F84F1000000        <1>         jz      K38
                             <1>       ;
000010E0 F6C710              <1>       test  bh, KBX                   ; IS THIS THE ENCHANCED KEYBOARD?
000010E3 740D                <1>       jz    short K29           ; NO, ALT STATE IS REAL
                             <1>        ;28/02/2015
000010E5 F605[8E5E0000]04    <1>       test  byte [KB_FLAG_1], SYS_SHIFT ; YES, IS SYSREQ KEY DOWN?
                             <1>       ;jz   short K29           ;  NO, ALT STATE IS REAL
000010EC 0F85DF000000        <1>       jnz   K38                 ; YES, THIS IS PHONY ALT STATE
                             <1>         ;                       ; DUE TO PRESSING SYSREQ
                             <1> ;K28A:      jmp    short K38
                             <1>       ;
                             <1>       ;----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
                             <1> K29:                            ; TEST-RESET
000010F2 F6C304              <1>       test  bl, CTL_SHIFT             ; ARE WE IN CONTROL SHIFT ALSO?
000010F5 740B                <1>       jz    short K31           ; NO_RESET
000010F7 3C53                <1>       cmp   al, DEL_KEY         ; CTL-ALT STATE, TEST FOR DELETE KEY
000010F9 7507                <1>       jne   short K31           ; NO_RESET, IGNORE
                             <1>       ;
                             <1>       ;----- CTL-ALT-DEL HAS BEEN FOUND
                             <1>       ; 26/08/2014
                             <1> cpu_reset:
                             <1>       ; IBM PC/AT ROM BIOS source code - 10/06/85 (TEST4.ASM - PROC_SHUTDOWN)
                             <1>       ; Send FEh (system reset command) to the keyboard controller.
000010FB B0FE                <1>       mov   al, SHUT_CMD        ; SHUTDOWN COMMAND
000010FD E664                <1>       out   STATUS_PORT, al           ; SEND TO KEYBOARD CONTROL PORT
                             <1> khere:
000010FF F4                  <1>       hlt                       ; WAIT FOR 80286 RESET
00001100 EBFD                <1>       jmp   short khere         ; INSURE HALT
                             <1>
                             <1>       ;
                             <1>       ;----- IN ALTERNATE SHIFT, RESET NOT FOUND
                             <1> K31:                            ; NO-RESET
00001102 3C39                <1>       cmp   al, 57              ; TEST FOR SPACE KEY
00001104 7507                <1>       jne   short K311          ; NOT THERE
00001106 B020                <1>       mov   al, ' '                   ; SET SPACE CHAR
00001108 E948020000          <1>         jmp     K57                     ; BUFFER_FILL
                             <1> K311:
0000110D 3C0F                <1>       cmp   al, 15              ; TEST FOR TAB KEY
```

```
0000110F 7509                <1>        jne    short K312          ; NOT THERE
00001111 66B800A5            <1>        mov    ax, 0A500h          ; SET SPECIAL CODE FOR ALT-TAB
00001115 E93B020000          <1>          jmp     K57                     ; BUFFER_FILL
                             <1> K312:
0000111A 3C4A                <1>        cmp    al, 74              ; TEST FOR KEY PAD -
0000111C 0F84A2000000        <1>          je      K37B                    ; GO PROCESS
00001122 3C4E                <1>        cmp    al, 78              ; TEST FOR KEY PAD +
00001124 0F849A000000        <1>          je      K37B                    ; GO PROCESS
                             <1>        ;
                             <1>        ;----- LOOK FOR KEY PAD ENTRY
                             <1> K32:                              ; ALT-KEY-PAD
0000112A BF[505D0000]        <1>        mov    edi, K30            ; ALT-INPUT-TABLE offset
0000112F B90A000000          <1>        mov    ecx, 10                   ; LOOK FOR ENTRY USING KEYPAD
00001134 F2AE                <1>        repne  scasb               ; LOOK FOR MATCH
00001136 7525                <1>        jne    short K33           ; NO_ALT_KEYPAD
00001138 F6C702              <1>        test   bh, LC_E0           ; IS THIS ONE OF THE NEW KEYS?
0000113B 0F858A000000        <1>          jnz     K37C                    ; YES, JUMP, NOT NUMPAD KEY
00001141 81EF[515D0000]      <1>        sub    edi, K30+1          ; DI NOW HAS ENTRY VALUE
00001147 A0[915E0000]        <1>        mov    al, [ALT_INPUT]     ; GET THE CURRENT BYTE
0000114C B40A                <1>        mov    ah, 10              ; MULTIPLY BY 10
0000114E F6E4                <1>        mul    ah
00001150 6601F8              <1>        add    ax, di              ; ADD IN THE LATEST ENTRY
00001153 A2[915E0000]        <1>        mov    [ALT_INPUT], al     ; STORE IT AWAY
                             <1> ;K32A:
00001158 E959FFFFFF          <1>          jmp     K26                     ; THROW AWAY THAT KEYSTROKE
                             <1>        ;
                             <1>        ;----- LOOK FOR SUPERSHIFT ENTRY
                             <1> K33:                              ; NO-ALT-KEYPAD
0000115D C605[915E0000]00    <1>          mov     byte [ALT_INPUT], 0     ; ZERO ANY PREVIOUS ENTRY INTO INPUT
00001164 B91A000000          <1>        mov    ecx, 26                   ; (DI),(ES) ALREADY POINTING
00001169 F2AE                <1>        repne  scasb               ; LOOK FOR MATCH IN ALPHABET
0000116B 7450                <1>        je     short K37A          ; MATCH FOUND, GO FILLL THE BUFFER
                             <1>        ;
                             <1>        ;----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
                             <1> K34:                              ; ALT-TOP-ROW
0000116D 3C02                <1>        cmp    al, 2               ; KEY WITH '1' ON IT
0000116F 7253                <1>        jb     short K37B          ; MUST BE ESCAPE
00001171 3C0D                <1>        cmp    al, 13              ; IS IT IN THE REGION
00001173 7705                <1>        ja     short K35           ; NO, ALT SOMETHING ELSE
00001175 80C476              <1>        add    ah, 118                   ; CONVERT PSEUDO SCAN CODE TO RANGE
00001178 EB43                <1>        jmp    short K37A          ; GO FILL THE BUFFER
                             <1>        ;
                             <1>        ;----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
                             <1> K35:                              ; ALT-FUNCTION
0000117A 3C57                <1>        cmp    al, F11_M           ; IS IT F11?
0000117C 7209                <1>        jb     short K35A ; 20/02/2015   ; NO, BRANCH
0000117E 3C58                <1>        cmp    al, F12_M           ; IS IT F12?
00001180 7705                <1>        ja     short K35A ; 20/02/2015   ; NO, BRANCH
00001182 80C434              <1>        add    ah, 52              ; CONVERT TO PSEUDO SCAN CODE
00001185 EB36                <1>        jmp    short K37A          ; GO FILL THE BUFFER
                             <1> K35A:
00001187 F6C702              <1>        test   bh, LC_E0           ; DO WE HAVE ONE OF THE NEW KEYS?
0000118A 7422                <1>        jz     short K37           ; NO, JUMP
0000118C 3C1C                <1>        cmp    al, 28              ; TEST FOR KEYPAD ENTER
0000118E 7509                <1>          jne     short K35B              ; NOT THERE
00001190 66B800A6            <1>        mov    ax, 0A600h          ; SPECIAL CODE
00001194 E9BC010000          <1>        jmp    K57                 ; BUFFER FILL
                             <1> K35B:
00001199 3C53                <1>        cmp    al, 83              ; TEST FOR DELETE KEY
0000119B 742E                <1>        je     short K37C          ; HANDLE WITH OTHER EDIT KEYS
0000119D 3C35                <1>        cmp    al, 53              ; TEST FOR KEYPAD /
                             <1>        ;jne   short K32A          ; NOT THERE, NO OTHER E0 SPECIALS
0000119F 0F8511FFFFFF        <1>          jne     K26
000011A5 66B800A4            <1>        mov    ax, 0A400h          ; SPECIAL CODE
000011A9 E9A7010000          <1>        jmp    K57                 ; BUFFER FILL
                             <1> K37:
000011AE 3C3B                <1>        cmp    al, 59              ; TEST FOR FUNCTION KEYS (F1)
000011B0 7212                <1>          jb      short K37B             ; NO FN, HANDLE W/OTHER EXTENDED
000011B2 3C44                <1>        cmp    al, 68              ; IN KEYPAD REGION?
                             <1>          ;ja  short K32A          ; IF SO, IGNORE
000011B4 0F87FCFEFFFF        <1>          ja      K26
000011BA 80C42D              <1>        add    ah, 45              ; CONVERT TO PSEUDO SCAN CODE
                             <1> K37A:
000011BD B000                <1>        mov    al, 0               ; ASCII CODE OF ZERO
000011BF E991010000          <1>          jmp     K57                     ; PUT IT IN THE BUFFER
                             <1> K37B:
000011C4 B0F0                <1>        mov    al, 0F0h            ; USE SPECIAL ASCII CODE
000011C6 E98A010000          <1>        jmp      K57                    ; PUT IT IN THE BUFFER
                             <1> K37C:
000011CB 0450                <1>        add    al, 80              ; CONVERT SCAN CODE (EDIT KEYS)
000011CD 88C4                <1>        mov    ah, al              ; (SCAN CODE NOT IN AH FOR INSERT)
000011CF EBEC                <1>        jmp      short K37A             ; PUT IT IN THE BUFFER
                             <1>        ;
                             <1>        ;----- NOT IN ALTERNATE SHIFT
                             <1> K38:                              ; NOT-ALT-SHIFT
                             <1>                                   ; BL STILL HAS SHIFT FLAGS
000011D1 F6C304              <1>        test   bl, CTL_SHIFT             ; ARE WE IN CONTROL SHIFT?
                             <1>        ;jnz   short K38A          ; YES, START PROCESSING
000011D4 0F84B0000000        <1>          jz      K44                     ; NOT-CTL-SHIFT
                             <1>        ;
                             <1>        ;----- CONTROL SHIFT, TEST SPECIAL CHARACTERS
                             <1>        ;----- TEST FOR BREAK
                             <1> K38A:
000011DA 3C46                <1>        cmp    al, SCROLL_KEY            ; TEST FOR BREAK
000011DC 7531                <1>        jne    short K39           ; JUMP, NO-BREAK
000011DE F6C710              <1>        test   bh, KBX                   ; IS THIS THE ENHANCED KEYBOARD?
000011E1 7405                <1>        jz     short K38B          ; NO, BREAK IS VALID
000011E3 F6C702              <1>        test   bh, LC_E0           ; YES, WAS LAST CODE AN E0?
000011E6 7427                <1>        jz     short K39           ; NO-BREAK, TEST FOR PAUSE
                             <1> K38B:
000011E8 8B1D[9A5E0000]      <1>        mov    ebx, [BUFFER_HEAD]  ; RESET BUFFER TO EMPTY
000011EE 891D[9E5E0000]      <1>        mov    [BUFFER_TAIL], ebx
000011F4 C605[8C5E0000]80    <1>        mov    byte [BIOS_BREAK], 80h  ; TURN ON BIOS_BREAK BIT
```

```
                             <1>       ;
                             <1>       ;----- ENABLE KEYBOARD
000011FB B0AE                <1>       mov    al, ENA_KBD         ; ENABLE KEYBOARD
000011FD E8BF010000          <1>       call   SHIP_IT                    ; EXECUTE ENABLE
                             <1>       ;
                             <1>       ; CTRL+BREAK code here !!!
                             <1>       ;INT   1BH                 ; BREAK INTERRUPT VECTOR
                             <1>       ; 17/10/2015
00001202 E8CF510000          <1>       call   ctrlbrk ; control+break subroutine
                             <1>       ;
00001207 6629C0              <1>       sub    ax, ax              ; PUT OUT DUMMY CHARACTER
0000120A E946010000          <1>         jmp     K57                     ; BUFFER_FILL
                             <1>       ;
                             <1>       ;----- TEST FOR PAUSE
                             <1> K39:                            ; NO_BREAK
0000120F F6C710              <1>       test   bh, KBX                    ; IS THIS THE ENHANCED KEYBOARD?
00001212 7537                <1>       jnz    short K41           ; YES, THEN THIS CAN'T BE PAUSE
00001214 3C45                <1>       cmp    al, NUM_KEY         ; LOOK FOR PAUSE KEY
00001216 7533                <1>       jne    short K41           ; NO-PAUSE
                             <1> K39P:
00001218 800D[8E5E0000]08    <1>       or     byte [KB_FLAG_1], HOLD_STATE ; TURN ON THE HOLD FLAG
                             <1>       ;
                             <1>       ;----- ENABLE KEYBOARD
0000121F B0AE                <1>       mov    al, ENA_KBD         ; ENABLE KEYBOARD
00001221 E89B010000          <1>       call   SHIP_IT                    ; EXECUTE ENABLE
                             <1> K39A:
00001226 B020                <1>       mov    al, EOI                    ; END OF INTERRUPT TO CONTROL PORT
00001228 E620                <1>       out    20h, al ;out INTA00, al    ; ALLOW FURTHER KEYSTROKE INTERRUPTS
                             <1>       ;
                             <1>       ;----- DURING PAUSE INTERVAL, TURN COLOR CRT BACK ON
0000122A 803D[C25E0000]07    <1>         cmp     byte [CRT_MODE], 7      ; IS THIS BLACK AND WHITE CARD
00001231 740A                <1>         je      short K40                    ; YES, NOTHING TO DO
00001233 66BAD803            <1>       mov    dx, 03D8h           ; PORT FOR COLOR CARD
00001237 A0[C35E0000]        <1>         mov     al, [CRT_MODE_SET]     ; GET THE VALUE OF THE CURRENT MODE
0000123C EE                  <1>       out    dx, al              ; SET THE CRT MODE, SO THAT CRT IS ON
                             <1>       ;
                             <1> K40:                            ; PAUSE-LOOP
0000123D F605[8E5E0000]08    <1>         test    byte [KB_FLAG_1], HOLD_STATE ; CHECK HOLD STATE FLAG
00001244 75F7                <1>       jnz    short K40           ; LOOP UNTIL FLAG TURNED OFF
                             <1>       ;
00001246 E977FEFFFF          <1>         jmp     K27                     ; INTERRUPT_RETURN_NO_EOI
                             <1>         ;
                             <1>       ;----- TEST SPECIAL CASE KEY 55
                             <1> K41:                            ; NO-PAUSE
0000124B 3C37                <1>       cmp    al, 55              ; TEST FOR */PRTSC KEY
0000124D 7513                <1>       jne    short K42           ; NOT-KEY-55
0000124F F6C710              <1>       test   bh, KBX                    ; IS THIS THE ENHANCED KEYBOARD?
00001252 7405                <1>       jz     short K41A          ; NO, CTL-PRTSC IS VALID
00001254 F6C702              <1>       test   bh, LC_E0           ; YES, WAS LAST CODE AN E0?
00001257 7421                <1>       jz     short K42B          ; NO, TRANSLATE TO A FUNCTION
                             <1> K41A:
00001259 66B80072            <1>       mov    ax, 114*256         ; START/STOP PRINTING SWITCH
0000125D E9F3000000          <1>         jmp     K57                     ; BUFFER_FILL
                             <1>       ;
                             <1>       ;----- SET UP TO TRANSLATE CONTROL SHIFT
                             <1> K42:                            ; NOT-KEY-55
00001262 3C0F                <1>       cmp    al, 15              ; IS IT THE TAB KEY?
00001264 7414                <1>       je     short K42B          ; YES, XLATE TO FUNCTION CODE
00001266 3C35                <1>       cmp    al, 53              ; IS IT THE / KEY?
00001268 750E                <1>       jne    short K42A          ; NO, NO MORE SPECIAL CASES
0000126A F6C702              <1>       test   bh, LC_E0           ; YES, IS IT FROM THE KEY PAD?
0000126D 7409                <1>       jz     short K42A          ; NO, JUST TRANSLATE
0000126F 66B80095            <1>       mov    ax, 9500h           ; YES, SPECIAL CODE FOR THIS ONE
00001273 E9DD000000          <1>       jmp    K57                 ; BUFFER FILL
                             <1> K42A:
                             <1>       ;;mov  ebx, _K8            ; SET UP TO TRANSLATE CTL
00001278 3C3B                <1>       cmp    al, 59              ; IS IT IN CHARACTER TABLE?
                             <1>         ;jb  short K45F             ; YES, GO TRANSLATE CHAR
                             <1>       ;;jb   K56 ; 20/02/2015
                             <1>       ;;jmp  K64 ; 20/02/2015
                             <1> K42B:
0000127A BB[845D0000]        <1>       mov    ebx, _K8            ; SET UP TO TRANSLATE CTL
0000127F 0F82AE000000        <1>       jb     K56 ;; 20/02/2015
00001285 E9B9000000          <1>       jmp    K64
                             <1>         ;
                             <1>       ;----- NOT IN CONTROL SHIFT
                             <1> K44:                            ; NOT-CTL-SHIFT
0000128A 3C37                <1>       cmp    al, 55              ; PRINT SCREEN KEY?
0000128C 7528                <1>       jne    short K45           ; NOT PRINT SCREEN
0000128E F6C710              <1>       test   bh, KBX                    ; IS THIS ENHANCED KEYBOARD?
00001291 7407                <1>       jz     short K44A          ; NO, TEST FOR SHIFT STATE
00001293 F6C702              <1>       test   bh, LC_E0           ; YES, LAST CODE A MARKER?
00001296 7507                <1>       jnz    short K44B          ; YES, IS PRINT SCREEN
00001298 EB41                <1>       jmp    short K45C          ; NO, TRANSLATE TO '*' CHARACTER
                             <1> K44A:
0000129A F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; NOT 101 KBD, SHIFT KEY DOWN?
0000129D 743C                <1>       jz     short K45C          ; NO, TRANSLATE TO '*' CHARACTER
                             <1>       ;
                             <1>       ;----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
                             <1> K44B:
0000129F B0AE                <1>       mov    al, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
000012A1 E81B010000          <1>       call   SHIP_IT                    ; EXECUTE ENABLE
000012A6 B020                <1>       mov    al, EOI                    ; END OF CURRENT INTERRUPT
000012A8 E620                <1>       out    20h, al ;out INTA00, al    ; SO FURTHER THINGS CAN HAPPEN
                             <1>       ; Print Screen !!!         ; ISSUE PRINT SCREEN INTERRUPT (INT 05h)
                             <1>       ;PUSH  BP                  ; SAVE POINTER
                             <1>       ;INT   5H                  ; ISSUE PRINT SCREEN INTERRUPT
                             <1>       ;POP   BP                  ; RESTORE POINTER
000012AA 8025[905E0000]FC    <1>         and     byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; ZERO OUT THESE FLAGS
000012B1 E90CFEFFFF          <1>         jmp     K27                     ; GO BACK WITHOUT EOI OCCURRING
                             <1>       ;
                             <1>       ;----- HANDLE IN-CORE KEYS
                             <1> K45:                            ; NOT-PRINT-SCREEN
```

```
000012B6 3C3A                <1>       cmp    al, 58             ; TEST FOR IN-CORE AREA
000012B8 7734                <1>       ja     short K46          ; JUMP IF NOT
000012BA 3C35                <1>       cmp    al, 53             ; IS THIS THE '/' KEY?
000012BC 7505                <1>       jne    short K45A         ; NO, JUMP
000012BE F6C702              <1>       test   bh, LC_E0          ; WAS THE LAST CODE THE MARKER?
000012C1 7518                <1>       jnz    short K45C         ; YES, TRANSLATE TO CHARACTER
                             <1> K45A:
000012C3 B91A000000          <1>       mov    ecx, 26                   ; LENGHT OF SEARCH
000012C8 BF[5A5D0000]        <1>       mov    edi, K30+10        ; POINT TO TABLE OF A-Z CHARS
000012CD F2AE                <1>       repne  scasb              ; IS THIS A LETTER KEY?
                             <1>              ; 20/02/2015
000012CF 7505                <1>       jne    short K45B               ; NO, SYMBOL KEY
                             <1>       ;
000012D1 F6C340              <1>       test   bl, CAPS_STATE            ; ARE WE IN CAPS_LOCK?
000012D4 750C                <1>       jnz    short K45D         ; TEST FOR SURE
                             <1> K45B:
000012D6 F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
000012D9 750C                <1>       jnz    short K45E         ; YES, UPPERCASE
                             <1>                                  ; NO, LOWERCASE
                             <1> K45C:
000012DB BB[DC5D0000]        <1>       mov    ebx, K10           ; TRANSLATE TO LOWERCASE LETTERS
000012E0 EB51                <1>       jmp    short K56
                             <1> K45D:                            ; ALMOST-CAPS-STATE
000012E2 F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; CL ON. IS SHIFT ON, TOO?
000012E5 75F4                <1>       jnz    short K45C         ; SHIFTED TEMP OUT OF CAPS STATE
                             <1> K45E:
000012E7 BB[345E0000]        <1>       mov    ebx, K11           ; TRANSLATE TO UPPER CASE LETTERS
000012EC EB45                <1> K45F: jmp    short K56
                             <1>       ;
                             <1>       ;----- TEST FOR KEYS F1 - F10
                             <1> K46:                             ; NOT IN-CORE AREA
000012EE 3C44                <1>       cmp    al, 68             ; TEST FOR F1 - F10
                             <1>       ;ja    short K47          ; JUMP IF NOT
                             <1>       ;jmp   short K53          ; YES, GO DO FN KEY PROCESS
000012F0 7635                <1>       jna    short K53
                             <1>       ;
                             <1>       ;----- HANDLE THE NUMERIC PAD KEYS
                             <1> K47:                             ; NOT F1 - F10
000012F2 3C53                <1>       cmp    al, 83             ; TEST NUMPAD KEYS
000012F4 772D                <1>       ja     short K52          ; JUMP IF NOT
                             <1>       ;
                             <1>       ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
                             <1> K48:
000012F6 3C4A                <1>       cmp    al , 74                   ; SPECIAL CASE FOR MINUS
000012F8 74ED                <1>       je     short K45E         ; GO TRANSLATE
000012FA 3C4E                <1>       cmp    al , 78                   ; SPECIAL CASE FOR PLUS
000012FC 74E9                <1>       je     short K45E         ; GO TRANSLATE
000012FE F6C702              <1>       test   bh, LC_E0          ; IS THIS ONE OFTHE NEW KEYS?
00001301 750A                <1>       jnz    short K49          ; YES, TRANSLATE TO BASE STATE
                             <1>       ;
00001303 F6C320              <1>       test   bl, NUM_STATE      ; ARE WE IN NUM LOCK
00001306 7514                <1>       jnz    short K50          ; TEST FOR SURE
00001308 F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
                             <1>       ;jnz   short K51          ; IF SHIFTED, REALLY NUM STATE
0000130B 75DA                <1>       jnz    short K45E
                             <1>       ;
                             <1>       ;----- BASE CASE FOR KEYPAD
                             <1> K49:
0000130D 3C4C                <1>       cmp    al, 76             ; SPECIAL CASE FOR BASE STATE 5
0000130F 7504                <1>       jne    short K49A         ; CONTINUE IF NOT KEYPAD 5
00001311 B0F0                <1>       mov    al, 0F0h           ; SPECIAL ASCII CODE
00001313 EB40                <1>       jmp    short K57          ; BUFFER FILL
                             <1> K49A:
00001315 BB[DC5D0000]        <1>       mov    ebx, K10           ; BASE CASE TABLE
0000131A EB27                <1>       jmp    short K64          ; CONVERT TO PSEUDO SCAN
                             <1>       ;
                             <1>       ;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
                             <1> K50:                             ; ALMOST-NUM-STATE
0000131C F6C303              <1>         test    bl, LEFT_SHIFT+RIGHT_SHIFT
0000131F 75EC                <1>       jnz    short K49          ; SHIFTED TEMP OUT OF NUM STATE
00001321 EBC4                <1> K51:  jmp    short K45E         ; REALLY NUM STATE
                             <1>       ;
                             <1>       ;----- TEST FOR THE NEW KEYS ON WT KEYBOARDS
                             <1> K52:                             ; NOT A NUMPAD KEY
00001323 3C56                <1>       cmp    al, 86             ; IS IT THE NEW WT KEY?
                             <1>       ;jne   short K53          ; JUMP IF NOT
                             <1>       ;jmp   short K45B         ; HANDLE WITH REST OF LETTER KEYS
00001325 74AF                <1>       je     short K45B
                             <1>       ;
                             <1>       ;----- MUST BE F11 OR F12
                             <1> K53:                             ; F1 - F10 COME HERE, TOO
00001327 F6C303              <1>       test   bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST SHIFT STATE
0000132A 74E1                <1>       jz     short K49          ; JUMP, LOWER CASE PSEUDO SC'S
                             <1>       ; 20/02/2015
0000132C BB[345E0000]        <1>       mov    ebx, K11           ; UPPER CASE PSEUDO SCAN CODES
00001331 EB10                <1>       jmp    short K64          ; TRANSLATE SCAN
                             <1>       ;
                             <1>       ;----- TRANSLATE THE CHARACTER
                             <1> K56:                             ; TRANSLATE-CHAR
00001333 FEC8                <1>       dec    al                 ; CONVERT ORIGIN
00001335 D7                  <1>       xlat                      ; CONVERT THE SCAN CODE TO ASCII
00001336 F605[905E0000]02    <1>       test   byte [KB_FLAG_3], LC_E0   ; IS THIS A NEW KEY?
0000133D 7416                <1>       jz     short K57          ; NO, GO FILL BUFFER
0000133F B4E0                <1>       mov    ah, MC_E0          ; YES, PUT SPECIAL MARKER IN AH
00001341 EB12                <1>       jmp    short K57          ; PUT IT INTO THE BUFFER
                             <1>       ;
                             <1>       ;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
                             <1> K64:                             ; TRANSLATE-SCAN-ORGD
00001343 FEC8                <1>       dec    al                 ; CONVERT ORIGIN
00001345 D7                  <1>              xlat                       ; CTL TABLE SCAN
00001346 88C4                <1>       mov    ah, al             ; PUT VALUE INTO AH
00001348 B000                <1>       mov    al, 0              ; ZERO ASCII CODE
0000134A F605[905E0000]02    <1>       test   byte [KB_FLAG_3], LC_E0   ; IS THIS A NEW KEY?
```

```
00001351 7402                 <1>       jz     short K57           ; NO, GO FILL BUFFER
00001353 B0E0                 <1>       mov    al, MC_E0           ; YES, PUT SPECIAL MARKER IN AL
                              <1>       ;
                              <1>       ;----- PUT CHARACTER INTO BUFFER
                              <1> K57:                             ; BUFFER_FILL
00001355 3CFF                 <1>       cmp    al, -1              ; IS THIS AN IGNORE CHAR
                              <1>         ;je  short K59           ; YES, DO NOTHING WITH IT
00001357 0F8459FDFFFF         <1>       je      K26                ; YES, DO NOTHING WITH IT
0000135D 80FCFF               <1>       cmp    ah, -1              ; LOOK FOR -1 PSEUDO SCAN
                              <1>         ;jne short K61           ; NEAR_INTERRUPT_RETURN
00001360 0F8450FDFFFF         <1>       je      K26                ; INTERRUPT_RETURN
                              <1> ;K59:                            ; NEAR_INTERRUPT_RETURN
                              <1> ;     jmp    K26                 ; INTERRUPT_RETURN
                              <1>
                              <1> _K60: ; 29/01/2016
00001366 80FC68               <1>       cmp    ah, 68h      ; ALT + F1 key
00001369 721F                 <1>       jb     short K61
0000136B 80FC6F               <1>       cmp    ah, 6Fh ; ALT + F8 key
0000136E 771A                 <1>       ja     short K61
                              <1>       ;
00001370 8A1D[66580100]       <1>       mov    bl, [ACTIVE_PAGE]
00001376 80C368               <1>       add    bl, 68h
00001379 38E3                 <1>       cmp    bl, ah
0000137B 740D                 <1>       je     short K61
0000137D 6650                 <1>       push   ax
0000137F 88E0                 <1>       mov    al, ah
00001381 2C68                 <1>       sub    al, 68h
00001383 E8F4050000           <1>       call   set_active_page
00001388 6658                 <1>       pop    ax
                              <1> K61:                             ; NOT-CAPS-STATE
0000138A 8B1D[9E5E0000]       <1>       mov    ebx, [BUFFER_TAIL]  ; GET THE END POINTER TO THE BUFFER
00001390 89DE                 <1>       mov    esi, ebx            ; SAVE THE VALUE
00001392 E857FAFFFF           <1>       call   _K4                 ; ADVANCE THE TAIL
00001397 3B1D[9A5E0000]       <1>       cmp    ebx, [BUFFER_HEAD]  ; HAS THE BUFFER WRAPPED AROUND
0000139D 740E                 <1>       je     short K62           ; BUFFER_FULL_BEEP
0000139F 668906               <1>       mov    [esi], ax           ; STORE THE VALUE
000013A2 891D[9E5E0000]       <1>       mov    [BUFFER_TAIL], ebx  ; MOVE THE POINTER UP
000013A8 E909FDFFFF           <1>       jmp    K26
                              <1>       ;;cli                      ; TURN OFF INTERRUPTS
                              <1>       ;;mov  al, EOI                    ; END OF INTERRUPT COMMAND
                              <1>       ;;out  INTA00, al          ; SEND COMMAND TO INTERRUPT CONTROL PORT
                              <1>       ;MOV   AL, ENA_KBD         ; INSURE KEYBOARD IS ENABLED
                              <1>       ;CALL  SHIP_IT                    ; EXECUTE ENABLE
                              <1>       ;MOV   AX, 9102H           ; MOVE IN POST CODE & TYPE
                              <1>       ;INT   15H                 ; PERFORM OTHER FUNCTION
                              <1>       ;;and  byte [KB_FLAG_3],~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
                              <1>       ;JMP   K27A                ; INTERRUPT_RETURN
                              <1>       ;;jmp   K27
                              <1>       ;
                              <1>       ;----- BUFFER IS FULL SOUND THE BEEPER
                              <1> K62:
000013AD B020                 <1>       mov    al, EOI                    ; ENABLE INTERRUPT CONTROLLER CHIP
000013AF E620                 <1>       out    INTA00, al
000013B1 66B9A602             <1>       mov    cx, 678                    ; DIVISOR FOR 1760 HZ
000013B5 B304                 <1>       mov    bl, 4               ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
000013B7 E8E5090000           <1>       call   beep                ; GO TO COMMON BEEP HANDLER
000013BC E901FDFFFF           <1>       jmp     K27                ; EXIT
                              <1>
                              <1> SHIP_IT:
                              <1>       ;---------------------------------------------------------------------------------
                              <1>       ; SHIP_IT
                              <1>       ;      THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
                              <1>       ;      TO THE KEYBOARD CONTROLLER.
                              <1>       ;---------------------------------------------------------------------------------
                              <1>       ;
000013C1 6650                 <1>       push   ax                  ; SAVE DATA TO SEND
                              <1>
                              <1>       ;----- WAIT FOR COMMAND TO ACCEPTED
000013C3 FA                   <1>       cli                        ; DISABLE INTERRUPTS TILL DATA SENT
                              <1>       ; xor  ecx, ecx            ; CLEAR TIMEOUT COUNTER
000013C4 B900000100           <1>       mov    ecx, 10000h
                              <1> S10:
000013C9 E464                 <1>       in     al, STATUS_PORT            ; READ KEYBOARD CONTROLLER STATUS
000013CB A802                 <1>       test   al, INPT_BUF_FULL   ; CHECK FOR ITS INPUT BUFFER BUSY
000013CD E0FA                 <1>       loopnz S10                 ; WAIT FOR COMMAND TO BE ACCEPTED
                              <1>
000013CF 6658                 <1>       pop    ax                  ; GET DATA TO SEND
000013D1 E664                 <1>       out    STATUS_PORT, al            ; SEND TO KEYBOARD CONTROLLER
000013D3 FB                   <1>       sti                        ; ENABLE INTERRUPTS AGAIN
000013D4 C3                   <1>       retn                       ; RETURN TO CALLER
                              <1>
                              <1> SND_DATA:
                              <1>       ; --------------------------------------------------------------------------------
                              <1>       ; SND_DATA
                              <1>       ;      THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
                              <1>       ;      TO THE KEYBOARD AND RECEIPT OF ACKNOWLEDGEMENTS. IT ALSO
                              <1>       ;      HANDLES ANY RETRIES IF REQUIRED
                              <1>       ; --------------------------------------------------------------------------------
                              <1>       ;
000013D5 6650                 <1>       push   ax                  ; SAVE REGISTERS
000013D7 6653                 <1>       push   bx
000013D9 51                   <1>       push   ecx
000013DA 88C7                 <1>       mov    bh, al              ; SAVE TRANSMITTED BYTE FOR RETRIES
000013DC B303                 <1>       mov    bl, 3               ; LOAD RETRY COUNT
                              <1> SD0:
000013DE FA                   <1>       cli                        ; DISABLE INTERRUPTS
000013DF 8025[8F5E0000]CF     <1>       and    byte [KB_FLAG_2], ~(KB_FE+KB_FA) ; CLEAR ACK AND RESEND FLAGS
                              <1>       ;
                              <1>       ;----- WAIT FOR COMMAND TO BE ACCEPTED
000013E6 B900000100           <1>       mov    ecx, 10000h         ; MAXIMUM WAIT COUNT
                              <1> SD5:
000013EB E464                 <1>       in     al, STATUS_PORT            ; READ KEYBOARD PROCESSOR STATUS PORT
000013ED A802                 <1>       test   al, INPT_BUF_FULL   ; CHECK FOR ANY PENDING COMMAND
```

```
000013EF E0FA                <1>       loopnz SD5                ; WAIT FOR COMMAND TO BE ACCEPTED
                             <1>       ;
000013F1 88F8                <1>       mov    al, bh             ; REESTABLISH BYTE TO TRANSMIT
000013F3 E660                <1>       out    PORT_A, al         ; SEND BYTE
000013F5 FB                  <1>       sti                       ; ENABLE INTERRUPTS
                             <1>       ;mov   cx, 01A00h         ; LOAD COUNT FOR 10 ms+
000013F6 B9FFFF0000          <1>       mov    ecx, 0FFFFh
                             <1> SD1:
000013FB F605[8F5E0000]30    <1>       test   byte [KB_FLAG_2], KB_FE+KB_FA ; SEE IF EITHER BIT SET
00001402 750F                <1>       jnz    short SD3          ; IF SET, SOMETHING RECEIVED GO PROCESS
00001404 E2F5                <1>       loop   SD1                ; OTHERWISE WAIT
                             <1> SD2:
00001406 FECB                <1>       dec    bl                 ; DECREMENT RETRY COUNT
00001408 75D4                <1>       jnz    short SD0          ; RETRY TRANSMISSION
0000140A 800D[8F5E0000]80    <1>       or     byte [KB_FLAG_2], KB_ERR ; TURN ON TRANSMIT ERROR FLAG
00001411 EB09                <1>       jmp    short SD4          ; RETRIES EXHAUSTED FORGET TRANSMISSION
                             <1> SD3:
00001413 F605[8F5E0000]10    <1>       test   byte [KB_FLAG_2], KB_FA ; SEE IF THIS IS AN ACKNOWLEDGE
0000141A 74EA                <1>       jz     short SD2          ; IF NOT, GO RESEND
                             <1> SD4:
0000141C 59                  <1>       pop    ecx                ; RESTORE REGISTERS
0000141D 665B                <1>       pop    bx
0000141F 6658                <1>       pop    ax
00001421 C3                  <1>       retn                      ; RETURN, GOOD TRANSMISSION
                             <1>
                             <1> SND_LED:
                             <1>       ; ---------------------------------------------------------------------------------
                             <1>       ; SND_LED
                             <1>       ;      THIS ROUTINES TURNS ON THE MODE INDICATORS.
                             <1>       ;
                             <1>       ;----------------------------------------------------------------------------------
                             <1>       ;
00001422 FA                  <1>       cli                       ; TURN OFF INTERRUPTS
00001423 F605[8F5E0000]40    <1>       test   byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
0000142A 755F                <1>       jnz    short SL1          ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
                             <1>       ;
0000142C 800D[8F5E0000]40    <1>       or     byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
00001433 B020                <1>       mov    al, EOI                  ; END OF INTERRUPT COMMAND
00001435 E620                <1>       out    20h, al ;out INTA00, al   ; SEND COMMAND TO INTERRUPT CONTROL PORT
00001437 EB11                <1>       jmp    short SL0          ; GO SEND MODE INDICATOR COMMAND
                             <1> SND_LED1:
00001439 FA                  <1>       cli                       ; TURN OFF INTERRUPTS
0000143A F605[8F5E0000]40    <1>       test   byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
00001441 7548                <1>       jnz    short SL1          ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
                             <1>       ;
00001443 800D[8F5E0000]40    <1>       or     byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
                             <1> SL0:
0000144A B0ED                <1>       mov    al, LED_CMD        ; LED CMD BYTE
0000144C E884FFFFFF          <1>       call   SND_DATA           ; SEND DATA TO KEYBOARD
00001451 FA                  <1>       cli
00001452 E836000000          <1>       call   MAKE_LED           ; GO FORM INDICATOR DATA BYTE
00001457 8025[8F5E0000]F8    <1>       and    byte [KB_FLAG_2], 0F8h    ; ~KB_LEDS ; CLEAR MODE INDICATOR BITS
0000145E 0805[8F5E0000]      <1>       or     [KB_FLAG_2], al     ; SAVE PRESENT INDICATORS FOR NEXT TIME
00001464 F605[8F5E0000]80    <1>       test   byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
0000146B 750F                <1>       jnz    short SL2          ; IF SO, BYPASS SECOND BYTE TRANSMISSION
                             <1>       ;
0000146D E863FFFFFF          <1>       call   SND_DATA           ; SEND DATA TO KEYBOARD
00001472 FA                  <1>       cli                       ; TURN OFF INTERRUPTS
00001473 F605[8F5E0000]80    <1>       test   byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
0000147A 7408                <1>       jz     short SL3          ; IF NOT, DON'T SEND AN ENABLE COMMAND
                             <1> SL2:
0000147C B0F4                <1>       mov    al, KB_ENABLE      ; GET KEYBOARD CSA ENABLE COMMAND
0000147E E852FFFFFF          <1>       call   SND_DATA           ; SEND DATA TO KEYBOARD
00001483 FA                  <1>       cli                       ; TURN OFF INTERRUPTS
                             <1> SL3:
00001484 8025[8F5E0000]3F    <1>       and    byte [KB_FLAG_2], ~(KB_PR_LED+KB_ERR) ; TURN OFF MODE INDICATOR
                             <1> SL1:                            ; UPDATE AND TRANSMIT ERROR FLAG
0000148B FB                  <1>       sti                       ; ENABLE INTERRUPTS
0000148C C3                  <1>       retn                      ; RETURN TO CALLER
                             <1>
                             <1> MAKE_LED:
                             <1>       ;---------------------------------------------------------------------------------
                             <1>       ; MAKE_LED
                             <1>       ;      THIS ROUTINES FORMS THE DATA BYTE NECESSARY TO TURN ON/OFF
                             <1>       ;      THE MODE INDICATORS.
                             <1>       ;---------------------------------------------------------------------------------
                             <1>       ;
                             <1>       ;push  cx                 ; SAVE CX
0000148D A0[8D5E0000]        <1>       mov    al, [KB_FLAG]      ; GET CAPS & NUM LOCK INDICATORS
00001492 2470                <1>       and    al, CAPS_STATE+NUM_STATE+SCROLL_STATE ; ISOLATE INDICATORS
                             <1>       ;mov   cl, 4              ; SHIFT COUNT
                             <1>       ;rol   al, cl             ; SHIFT BITS OVER TO TURN ON INDICATORS
00001494 C0C004              <1>       rol    al, 4 ; 20/02/2015
00001497 2407                <1>       and    al, 07h                   ; MAKE SURE ONLY MODE BITS ON
                             <1>       ;pop   cx
00001499 C3                  <1>       retn                      ; RETURN TO CALLER
                             <1>
                             <1> ; % include 'kybdata.s'   ; KEYBOARD DATA
                             <1>
                             <1>
                             <1> ; /// End Of KEYBOARD FUNCTIONS ///

                                 %include 'video.s' ; 07/03/2015
                             <1> ; ****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - video.s
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 09/12/2017
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 16/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Turkish Rational DOS
                             <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
                             <1> ;
                             <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                             <1> ; video.inc (13/08/2015)
                             <1> ;
                             <1> ; Derived from 'IBM PC-AT' BIOS source code (1985)
                             <1> ; *****************************************************************************
                             <1>
                             <1> ; Retro UNIX 386 v1 Kernel - VIDEO.INC
                             <1> ; Last Modification: 13/08/2015
                             <1> ;              (Video Data is in 'VIDATA.INC')
                             <1> ;
                             <1> ; ///////// VIDEO (CGA) FUNCTIONS ///////////////
                             <1>
                             <1> ; 16/01/2016 (32 bit modifications, TRDOS386 - TRDOS v2.0, video.s)
                             <1> ; INT 31H (TRDOS 386) = INT 10H (IBM PC/AT REAL MODE)
                             <1>
                             <1> ; IBM PC-AT BIOS Source Code
                             <1> ; TITLE VIDEO1 --- 06/10/85  VIDEO DISPLAY BIOS
                             <1>
                             <1> _int10h:
                             <1>       ; 23/03/2016
                             <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
0000149A 9C                  <1>       pushfd
0000149B 0E                  <1>       push   cs
0000149C E851000000          <1>       call   VIDEO_IO_1
000014A1 C3                  <1>       retn
                             <1>
                             <1> ;--- INT 10 H --------------------------------------------------------------------
                             <1> ; VIDEO_IO                                                        :
                             <1> ;     THESE ROUTINES PROVIDE THE CRT DISPLAY INTERFACE                  :
                             <1> ;     THE FOLLOWING FUNCTIONS ARE PROVIDED:                             :
                             <1> ;                                                                 :
                             <1> ;    (AH)= 00H     SET MODE (AL) CONTAINS MODE VALUE                           :
                             <1> ;           (AL) = 00H  40X25 BW MODE (POWER ON DEFAULT)               :
                             <1> ;           (AL) = 01H  40X25 COLOR                                    :
                             <1> ;           (AL) = 02H  80X25 BW                                       :
                             <1> ;           (AL) = 03H  80X25 COLOR                                    :
                             <1> ;                         GRAPHICS MODES                               :
                             <1> ;           (AL) = 04H  320X200 COLOR                            :
                             <1> ;           (AL) = 05H  320X200 BW MODE                                :
                             <1> ;           (AL) = 06H  640X200 BW MODE                                :
                             <1> ;           (AL) = 07H   80X25 MONOCHROME (USED INTERNAL TO VIDEO ONLY)       :
                             <1> ;           *** NOTES -BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR :
                             <1> ;                      BURST IS NOT ENABLED                            :
                             <1> ;                     -CURSOR IS NOT DISPLAYED IN GRAPHICS MODE        :
                             <1> ;    (AH)= 01H     SET CURSOR TYPE                                             :
                             <1> ;           (CH) = BITS 4-0 = START LINE FOR CURSOR                    :
                             <1> ;                  ** HARDWARE WILL ALWAYS CAUSE BLINK                 :
                             <1> ;                  ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING   :
                             <1> ;                     OR NO CURSOR AT ALL                              :
                             <1> ;           (CL) = BITS 4-0 = END LINE FOR CURSOR                      :
                             <1> ;    (AH)= 02H     SET CURSOR POSITION                                 :
                             <1> ;           (DH,DL) = ROW,COLUMN  (00H,00H) IS UPPER LEFT              :
                             <1> ;           (BH) = A PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)             :
                             <1> ;    (AH)= 03H     READ CURSOR POSITION                                        :
                             <1> ;           (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES)        :
                             <1> ;           ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR                    :
                             <1> ;                   (CH,CL) = CURSOR MODE CURRENTLY SET                :
                             <1> ;    (AH)= 04H     READ LIGHT PEN POSITION                                     :
                             <1> ;           ON EXIT:                                              :
                             <1> ;           (AH) = 00H -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED             :
                             <1> ;           (AH) = 01H -- VALID LIGHT PEN VALUE IN REGISTERS           :
                             <1> ;                   (DH,DL) = ROW,COLUMN OF CHARACTER LP POSITION             :
                             <1> ;                   (CH) = RASTER LINE (0-199)                         :
                             <1> ;                   (BX) = PIXEL COLUMN (0-319,639)                    :
                             <1> ;    (AH)= 05H     SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES)           :
                             <1> ;           (AL) = NEW PAGE VALUE (0-7 FOR MODES 0&1, 0-3 FOR MODES 2&3)      :
                             <1> ;    (AH)= 06H     SCROLL ACTIVE PAGE UP                                       :
                             <1> ;           (AL) = NUMBER OF LINES. ( LINES BLANKED AT BOTTOM OF WINDOW )     :
                             <1> ;                   (AL) = 00H MEANS BLANK ENTIRE WINDOW               :
                             <1> ;           (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL        :
                             <1> ;           (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL       :
                             <1> ;           (BH) = ATTRIBUTE TO BE USED ON BLANK LINE                  :
                             <1> ;    (AH)= 07H     SCROLL ACTIVE PAGE DOWN                                     :
                             <1> ;           (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW      :
                             <1> ;                   (AL) = 00H MEANS BLANK ENTIRE WINDOW               :
                             <1> ;           (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL        :
                             <1> ;           (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL       :
                             <1> ;           (BH) = ATTRIBUTE TO BE USED ON BLANK LINE                  :
                             <1> ;                                                                 :
                             <1> ;   CHARACTER HANDLING ROUTINES                                        :
                             <1> ;                                                                 :
                             <1> ;    (AH)= 08H     READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION         :
                             <1> ;           (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)           :
                             <1> ;           ON EXIT:                                              :
                             <1> ;           (AL) = CHAR READ                                      :
                             <1> ;           (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY)             :
                             <1> ;    (AH)= 09H     WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION        :
                             <1> ;           (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)           :
                             <1> ;           (CX) = COUNT OF CHARACTERS TO WRITE                        :
                             <1> ;           (AL) = CHAR TO WRITE                                       :
                             <1> ;           (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS)    :
                             <1> ;                    SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.        :
                             <1> ;    (AH) = 0AH    WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION                   :
                             <1> ;           (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)           :
                             <1> ;           (CX) = COUNT OF CHARACTERS TO WRITE                        :
                             <1> ;           (AL) = CHAR TO WRITE                                       :
                             <1> ;                  NOTE: USE FUNCTION (AH)= 09H IN GRAPHICS MODES             :
                             <1> ;     FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE          :
                             <1> ;           CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE            :
                             <1> ;           MAINTAINED IN THE SYSTEM ROM. ONLY THE 1ST 128 CHARS       :
                             <1> ;           ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128 CHARS,   :
                             <1> ;           THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH           :
                             <1> ;           (LOCATION 0007CH) TO POINT TO THE 1K BYTE TABLE CONTAINING :
                             <1> ;           THE CODE POINTS FOR THE SECOND 128 CHARS (128-255).        :
                             <1> ;     FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR  :
                             <1> ;           CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY :
                             <1> ;           FOR CHARACTERS CONTAINED ON THE SAME ROW. CONTINUATION TO  :
                             <1> ;           SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.               :
                             <1> ;                                                            :
                             <1> ;    GRAPHICS INTERFACE                                              :
                             <1> ;    (AH)= 0BH     SET COLOR PALETTE                                 :
                             <1> ;           (BH) = PALETTE COLOR ID BEING SET (0-127)                :
                             <1> ;           (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID         :
                             <1> ;                  NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT HAS     :
                             <1> ;                          MEANING ONLY FOR 320X200 GRAPHICS.        :
                             <1> ;                  COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15)           :
                             <1> ;                  COLOR ID = 1 SELECTS THE PALETTE TO BE USED:      :
                             <1> ;                          0 = GREEN(1)/RED(2)/YELLOW(3)             :
                             <1> ;                          1 = CYAN(1)/MAGENTA(2)/WHITE(3)                    :
                             <1> ;                  IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET FOR  :
                             <1> ;                          PALETTE COLOR 0 INDICATES THE BORDER COLOR :
                             <1> ;                          TO BE USED (VALUES 0-31, WHERE 16-31 SELECT :
                             <1> ;                          THE HIGH INTENSITY BACKGROUND SET.        :
                             <1> ;    (AH)= 0CH     WRITE DOT                                         :
                             <1> ;           (DX) = ROW NUMBER                                  :
                             <1> ;           (CX) = COLUMN NUMBER                                     :
                             <1> ;           (AL) = COLOR VALUE                                 :
                             <1> ;                   IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE     :
                             <1> ;                   ORed WITH THE CURRENT CONTENTS OF THE DOT        :
                             <1> ;    (AH)= ODH     READ DOT                                          :
                             <1> ;           (DX) = ROW NUMBER                                  :
                             <1> ;           (CX) = COLUMN NUMBER                                     :
                             <1> ;           (AL) = RETURNS THE DOT READ                              :
                             <1> ;                                                            :
                             <1> ;    ASCII TELETYPE ROUTINE FOR OUTPUT                               :
                             <1> ;                                                            :
                             <1> ;    (AH)= 0EH     WRITE TELETYPE TO ACTIVE PAGE                              :
                             <1> ;           (AL) = CHAR TO WRITE                                     :
                             <1> ;           (BL) = FOREGROUND COLOR IN GRAPHICS MODE                 :
                             <1> ;           NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET           :
                             <1> ;    (AH)= 0FH     CURRENT VIDEO STATE                               :
                             <1> ;           RETURNS THE CURRENT VIDEO STATE                          :
                             <1> ;           (AL) = MODE CURRENTLY SET ( SEE (AH)=00H FOR EXPLANATION) :
                             <1> ;           (AH) = NUMBER OR CHARACTER COLUMNS ON SCREEN             :
                             <1> ;           (BH) = CURRENT ACTIVE DISPLAY PAGE                       :
                             <1> ;    (AH)= 10H     RESERVED                                          :
                             <1> ;    (AH)= 11H     RESERVED                                          :
                             <1> ;    (AH)= 12H     RESERVED                                          :
                             <1> ;    (AH)= 13H     WRITE STRING                                      :
                             <1> ;                  ES:BP  -  POINTER T0 STRING TO BE WRITTEN         :
                             <1> ;                  CX     -  LENGTH OF CHARACTER STRING TO WRITTEN            :
                             <1> ;                  DX     -  CURSOR POSITION FOR STRING TO BE WRITTEN :
                             <1> ;                  BH     -  PAGE NUMBER                             :
                             <1> ;           (AL)= 00H    WRITE CHARACTER STRING                      :
                             <1> ;                  BL     -  ATTRIBUTE                         :
                             <1> ;                  STRING IS  <CHAR,CHAR, ... ,CHAR>                 :
                             <1> ;                  CURSOR NOT MOVED                            :
                             <1> ;           (AL)= 01H    WRITE CHARACTER STRING AND MOVE CURSOR      :
                             <1> ;                  BL     -  ATTRIBUTE                         :
                             <1> ;                  STRING IS  <CHAR,CHAR, ... ,CHAR>                 :
                             <1> ;                  CURSOR MOVED                                :
                             <1> ;           (AL)= 02H    WRITE CHARACTER AND ATTRIBUTE STRING        :
                             <1> ;                         (VALID FOR ALPHA MODES ONLY)               :
                             <1> ;                  STRING IS <CHAR,ATTR,CHAR,ATTR ..  ,CHAR,ATTR>             :
                             <1> ;                  CURSOR IS NOT MOVED                         :
                             <1> ;           (AL)= 03H WRITE CHARACTER AND ATTRIBUTE STRING AND MOVE CURSOR   :
                             <1> ;                         (VALID FOR ALPHA MODES ONLY)               :
                             <1> ;                  STRING IS <CHAR,ATTR,CHAR,ATTR ..  ,CHAR,ATTR>             :
                             <1> ;                  CURSOR IS MOVED                                   :
                             <1> ;            NOTE:  CARRIAGE RETURN, LINE FEED, BACKSPACE, AND BELL ARE       :
                             <1> ;                   TREATED AS COMMANDS RATHER THAN PRINTABLE CHARACTERS.     :
                             <1> ;                                                            :
                             <1> ;     BX,CX,DX,SI,DI,BP,SP,DS,ES,SS PRESERVED DURING CALLS EXCEPT FOR         :
                             <1> ;     BX,CX,DX RETURN VALUES ON FUNCTIONS 03H,04H,0DH AND 0FH. ON ALL CALLS   :
                             <1> ;     AX IS MODIFIED.                                                :
                             <1> ;--------------------------------------------------------------------------------
                             <1>
000014A2 [4F150000]           <1> M1:   dd     SET_MODE     ; TABLE OF ROUTINES WITHIN VIDEO I/O
000014A6 [B7180000]           <1>       dd     SET_CTYPE
000014AA [EB180000]           <1>       dd     SET_CPOS
000014AE [13190000]           <1>       dd     READ_CURSOR
                             <1>       ;dd    VIDEO_RETURN ; READ_LPEN
000014B2 [38150000]           <1>       dd     set_mode_ncm ; Set mode without clearing video memory
000014B6 [59190000]           <1>       dd     ACT_DISP_PAGE
000014BA [F0190000]           <1>       dd     SCROLL_UP
000014BE [141B0000]           <1>       dd     SCROLL_DOWN
000014C2 [951B0000]           <1>       dd     READ_AC_CURRENT
000014C6 [ED1B0000]           <1>       dd     WRITE_AC_CURRENT
000014CA [131C0000]           <1>       dd     WRITE_C_CURRENT
000014CE [39250000]           <1>       dd     SET_COLOR
000014D2 [A4250000]           <1>       dd     WRITE_DOT
000014D6 [6F250000]           <1>       dd     READ_DOT
000014DA [951C0000]           <1>       dd     WRITE_TTY
000014DE [20150000]           <1>       dd     VIDEO_STATE
000014E2 [EF2E0000]           <1>       dd     vga_pal_funcs ; 10/08/2016 (TRDOS 386)
000014E6 [A52A0000]           <1>       dd     font_setup    ; 10/07/2016 (TRDOS 386)
000014EA [54150000]           <1>       dd     VIDEO_RETURN ; RESERVED
000014EE [021E0000]           <1>       dd     WRITE_STRING  ; 23/06/2016 (TRDOS 386)
                             <1> M1L   EQU    $ - M1
                             <1>
                             <1> ; 14/01/2017
                             <1> ; 02/01/2017
                             <1> ; 04/07/2016
                             <1> ; 12/05/2016
```

```
                             <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> int31h:  ; Video BIOS
                             <1>
                             <1> ; BH = Video page number
                             <1> ; BL = Color/Attribute
                             <1> ; AH = Function number
                             <1> ; AL = Character
                             <1>
                             <1> VIDEO_IO_1:
                             <1>       ;sti                      ; INTERRUPTS BACK ON
000014F2 FC                  <1>       cld                       ; SET DIRECTION FORWARD
000014F3 80FC14              <1>       cmp    ah, M1L/4          ; TEST FOR WITHIN TABLE RANGE
000014F6 7327                <1>       jnb    short M4           ; BRANCH TO EXIT IF NOT A VALID COMMAND
                             <1>
000014F8 06                  <1>       push   es
000014F9 1E                  <1>       push   ds                 ; SAVE WORK AND PARAMETER REGISTERS
000014FA 52                  <1>       push   edx
000014FB 51                  <1>       push   ecx
000014FC 53                  <1>       push   ebx
000014FD 56                  <1>       push   esi
000014FE 57                  <1>       push   edi
000014FF 55                  <1>       push   ebp
                             <1>
00001500 66BE1000            <1>       mov    si, KDATA          ; POINT DS: TO DATA SEGMENT
00001504 8EDE                <1>       mov    ds, si
00001506 8EC6                <1>       mov    es, si
00001508 BF00800B00          <1>       mov    edi, 0B8000h       ; GET offset FOR COLOR CARD
0000150D A3[C4650100]        <1>       mov    [video_eax], eax ; 12/05/2016
                             <1>       ; 23/03/2016
00001512 C0E402              <1>       shl    ah, 2  ; dword            ; TIMES 2 FOR WORD TABLE LOOKUP
00001515 0FB6F4              <1>       movzx esi, ah                    ; MOVE OFFSET INTO LOOK UP REGISTER (SI)
                             <1>       ;mov   ah, [CRT_MODE]            ; MOVE CURRENT MODE INTO (AH) REGISTER
                             <1>
                             <1>       ;;15/01/2017
                             <1>       ; 14/01/2017
                             <1>       ; 02/01/2017
                             <1>       ;;mov  byte [intflg], 31h  ; video interrupt
00001518 FB                  <1>       sti
                             <1>       ;
                             <1>
00001519 FFA6[A2140000]      <1>       JMP    dword [esi+M1]            ; GO TO SELECTED FUNCTION
                             <1>
                             <1> M4:                               ; COMMAND NOT VALID
0000151F CF                  <1>       iretd                     ; DO NOTHING IF NOT IN VALID RANGE
                             <1>
                             <1> VIDEO_STATE:
                             <1>       ; 26/06/2016
                             <1>       ; 12/05/2016
                             <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>
                             <1> ;---------------------------------------------------
                             <1> ; VIDEO STATE
                             <1> ;  RETURNS THE CURRENT VIDEO STATE IN AX
                             <1> ;  AH = NUMBER OF COLUMNS ON THE SCREEN
                             <1> ;  AL = CURRENT VIDEO MODE
                             <1> ;  BH = CURRENT ACTIVE PAGE
                             <1> ;---------------------------------------------------
                             <1>
00001520 8A25[C45E0000]      <1>       mov    ah, [CRT_COLS]     ; GET NUMBER OF COLUMNS
00001526 A0[C25E0000]        <1>       mov    al, [CRT_MODE]     ; CURRENT MODE
                             <1>       ;movzx esi, al
                             <1>       ;mov   ah, [esi+M6]
                             <1>       ; BH = active page
0000152B 8A3D[66580100]      <1>       mov    bh, [ACTIVE_PAGE] ; GET CURRENT ACTIVE PAGE
00001531 FA                  <1>       cli    ; 02/01/2017
00001532 5D                  <1>       pop    ebp          ; RECOVER REGISTERS
00001533 5F                  <1>       pop    edi
00001534 5E                  <1>       pop    esi
00001535 59                  <1>       pop    ecx          ; DISCARD SAVED BX
00001536 EB26                <1>       jmp    short M15    ; RETURN TO CALLER
                             <1>
                             <1> set_mode_ncm:
                             <1>       ; 04/07/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; set mode without clearing the video memory
                             <1>       ; (ony for graphics modes)
00001538 3C07                <1>       cmp    al, 7 ; IBM PC CGA modes
0000153A 7613                <1>       jna    short SET_MODE ; normal function (clear)
                             <1>       ; do not clear memory
0000153C A2[D3650100]        <1>       mov    [noclearmem], al ; > 0
00001541 E81F000000          <1>       call   _set_mode
00001546 C605[D3650100]00    <1>       mov    byte [noclearmem], 0
0000154D EB05                <1>         jmp     short VIDEO_RETURN
                             <1>
                             <1>       ; 10/08/2016
                             <1>       ; 08/08/2016
                             <1>       ; 30/07/2016
                             <1>       ; 29/07/2016
                             <1>       ; 27/07/2016
                             <1>       ; 26/07/2016
                             <1>       ; 25/07/2016
                             <1>       ; 23/07/2016
                             <1>       ; 18/07/2016
                             <1>       ; 02/07/2016
                             <1>       ; 26/06/2016
                             <1>       ; 24/06/2016
                             <1>       ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> SET_MODE:
                             <1>       ; For 32 bit TRDOS and Retro UNIX 386:
                             <1>       ;      valid video mode: 03h only!
                             <1>       ;      (VGA modes will be selected with another routine)
                             <1>       ;
                             <1>       ; set_txt_mode ; 80*25 (16 fore colors, 8 back colors)
                             <1>
                             <1> ;------------------------------------------------------
                             <1> ; SET MODE                                   :
                             <1> ;     THIS ROUTINE INITIALIZES THE ATTACHMENT TO    :
                             <1> ;     THE SELECTED MODE, THE SCREEN IS BLANKED.     :
                             <1> ; INPUT                                            :
                             <1> ;     (AL) - MODE SELECTED (RANGE 0-7)       :
                             <1> ; OUTPUT                                     :
                             <1> ;     NONE                                   :
                             <1> ;------------------------------------------------------
                             <1> 
0000154F E811000000          <1>       call   _set_mode ; 24/06/2016 (set_txt_mode)
                             <1> 
                             <1> ; 12/05/2016
                             <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1> 
                             <1> ;-----      NORMAL RETURN FROM ALL VIDEO RETURNS
                             <1> 
                             <1> VIDEO_RETURN:
00001554 A1[C4650100]        <1>       mov    eax, [video_eax] ; 12/05/2016
                             <1> _video_return:
00001559 FA                  <1>       cli ; 02/01/2017
0000155A 5D                  <1>       pop    ebp
0000155B 5F                  <1>       pop    edi
0000155C 5E                  <1>       pop    esi
0000155D 5B                  <1>       pop    ebx
                             <1> M15:  ; VIDEO_RETURN_C
                             <1>       ;;15/01/2017
                             <1>       ; 02/01/2017
                             <1>       ;;mov  byte [intflg], 0
                             <1>       ;
0000155E 59                  <1>       pop    ecx
0000155F 5A                  <1>       pop    edx
00001560 1F                  <1>       pop    ds
00001561 07                  <1>       pop    es     ; RECOVER SEGMENTS
00001562 CF                  <1>       iretd        ; ALL DONE
                             <1> 
                             <1> set_txt_mode:
                             <1>       ; 29/07/2016
                             <1>       ; 27/06/2016
00001563 B003                <1>       mov    al, 3
                             <1> 
                             <1> ; 10/08/2016
                             <1> ; 08/08/2016
                             <1> ; 30/07/2016
                             <1> ; 29/07/2016
                             <1> ; 27/07/2016
                             <1> ; 26/07/2016
                             <1> ; 25/07/2016
                             <1> ; 23/07/2016
                             <1> ; 18/07/2016
                             <1> ; 07/07/2016
                             <1> ; 04/07/2016
                             <1> ; 03/07/2016
                             <1> ; 02/07/2016
                             <1> ; 26/06/2016
                             <1> ; 24/06/2016 (set_txt_mode -> _set_mode)
                             <1> ; 17/06/2016
                             <1> ; 29/05/2016
                             <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1> _set_mode:
                             <1>       ; 24/06/2016
00001565 3805[C25E0000]      <1>       cmp    [CRT_MODE], al  ; current mode = requested mode ?
0000156B 750D                <1>       jne    short _sm_0
0000156D 3C03                <1>       cmp    al, 3        ; text, 80*25 color, default mode
                             <1>                          ; for TRDOS 386 MainProg
0000156F 755F                <1>         jne     short _sm_2     ;  multiscreen is only for mode 3
                             <1> 
                             <1>       ; If '_set_mode' procedure is called for video mode 3
                             <1>       ;     while video mode is 3, video page will be cleared
                             <1>       ;     and cursor position of video page will be reset.
                             <1> 
                             <1>       ; 29/07/2016
00001571 800D[D1650100]80    <1>       or     byte [p_crt_mode], 80h ; clear page indicator
00001578 EB5B                <1>         jmp     short _sm_3
                             <1> _sm_0:
0000157A 803D[C25E0000]03    <1>       cmp    byte [CRT_MODE], 3
00001581 7534                <1>         jne     short _sm_1
                             <1> 
                             <1>       ; If '_set_mode' procedure is called for a video mode
                             <1>       ;     except video mode 3, while current video mode
                             <1>       ;     is 3; all video pages of mode 3 will be copied
                             <1>       ;     to 98000h address as backup, before mode change.
                             <1> 
                             <1> _sm_save_pm:
                             <1>       ; 03/07/2016
                             <1>       ; save video pages
00001583 BE00800B00          <1>       mov    esi, 0B8000h
00001588 BF00800900          <1>       mov    edi, 98000h ; 30/07/2016
0000158D B900200000          <1>       mov    ecx, (0B8000h-0B0000h)/4
00001592 F3A5                <1>       rep    movsd
                             <1> 
00001594 C605[D1650100]03    <1>       mov    byte [p_crt_mode], 3 ; previous mode, backup sign
                             <1>       ;mov   cl, [ACTIVE_PAGE]
                             <1>       ;mov   [p_crt_page], cl
                             <1> 
                             <1>       ; save cursor positions
0000159B BE[56580100]        <1>       mov    esi, CURSOR_POSN
000015A0 BF[D6650100]        <1>       mov    edi, cursor_pposn    ; cursor positions backup
000015A5 B104                <1>       mov    cl, 4
000015A7 F3A5                <1>       rep    movsd
                             <1> 
                             <1>       ; 29/07/2016
                             <1>       ;mov   [ACTIVE_PAGE], cl ; 0
```

```
000015A9 860D[66580100]      <1>       xchg   cl, [ACTIVE_PAGE]
000015AF 880D[D2650100]      <1>       mov    [p_crt_page], cl     ; previous page (for mode 3)
                             <1>       ; [ACTIVE_PAGE] = 0
000015B5 EB19                <1>       jmp    short _sm_2
                             <1> 
                             <1> _sm_1:
000015B7 3C03                <1>       cmp    al, 3        ; text, 80*25 color, default mode
                             <1>                           ; for TRDOS 386 MainProg
000015B9 7515                <1>       jne    short _sm_2  ;  multiscreen is only for mode 3
                             <1> 
                             <1>       ; If '_set_mode' procedure is called for video mode 3
                             <1>       ;     while video mode is not 3 and if there is video
                             <1>       ;     page backup for video mode 3, all (of 8) mode 3
                             <1>       ;     video pages will be restored from 98000h.
                             <1> 
000015BB 803D[D1650100]03    <1>       cmp    byte [p_crt_mode], 3 ; previous mode, backup sign
000015C2 750C                <1>       jne    short _sm_2 ; there is no (multiscreen) video pages
                             <1>                      ; to be restored
000015C4 8A0D[D2650100]      <1>       mov    cl, [p_crt_page]
000015CA 880D[66580100]      <1>       mov    [ACTIVE_PAGE], cl
                             <1> 
                             <1> _sm_2:
000015D0 A2[C25E0000]        <1>       mov    [CRT_MODE], al  ; save mode in global variable
                             <1> _sm_3:
                             <1>       ; 30/07/2016
                             <1>       ; 26/07/2016
                             <1>       ; 25/07/2016
                             <1>       ; set_mode_vga:
                             <1>       ; 18/07/2016
                             <1>       ; 14/07/2016
                             <1>       ; 09/07/2016
                             <1>       ; 04/07/2016
                             <1>       ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; /// video mode 13h ///
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'vgatables.h'
                             <1>       ;
                             <1>       ; Oracle VirtualBox 5.0.24 VGABios Source Code
                             <1>       ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
                             <1>       ;
000015D5 88C4                <1>       mov    ah, al
000015D7 B910000000          <1>       mov    ecx, vga_mode_count
000015DC BE[DE5E0000]        <1>       mov    esi, vga_modes
000015E1 31DB                <1>       xor    ebx, ebx
                             <1> _sm_4:
000015E3 AC                  <1>       lodsb
000015E4 38C4                <1>       cmp    ah, al
000015E6 740C                <1>       je     short _sm_5
000015E8 FEC3                <1>       inc    bl
000015EA E2F7                <1>       loop   _sm_4
                             <1> 
                             <1>       ; UNIMPLEMENTED VIDEO MODE !
000015EC 31C0                <1>       xor    eax, eax
000015EE A3[C4650100]        <1>         mov     [video_eax], eax ; 0
000015F3 C3                  <1>       retn
                             <1> 
                             <1> ;-----      eBX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
                             <1> 
                             <1> _sm_5:      ; 25/07/2016
000015F4 89DE                <1>       mov    esi, ebx
000015F6 81C6[2E5F0000]      <1>       add    esi, vga_memmodel
000015FC 8A06                <1>       mov    al, [esi]
000015FE A2[EA650100]        <1>       mov    [VGA_MTYPE], al
                             <1> 
00001603 89DF                <1>       mov    edi, ebx
00001605 81C7[3E5F0000]      <1>       add    edi, vga_dac_s
0000160B C0E302              <1>       shl    bl, 2 ; byte -> dword
0000160E 81C3[EE5E0000]      <1>       add    ebx, vga_mode_tbl_ptr
                             <1> 
                             <1>       ;mov   dword [VGA_BASE], 0B8000h
                             <1>       ;cmp   ah, 0Dh ; [CRT_MODE]
                             <1>       ;jb    short M9
                             <1>       ;mov   dword [VGA_BASE], 0A0000h
                             <1> ;M9:
00001614 8B33                <1>       mov    esi, [ebx]
00001616 89F3                <1>       mov    ebx, esi
00001618 83C614              <1>       add    esi, vga_p_cm_pos ; ebx + 20
0000161B 668B06              <1>       mov    ax, [esi]       ; get the cursor mode from the table
0000161E 66A3[DB5E0000]      <1>       mov    [CURSOR_MODE], ax ; save cursor mode (initial value)
                             <1>       ; al = 6, ah = 7
                             <1>       ; al = 0Dh, ah = 0Eh ; 25/07/2016
00001624 E83B020000          <1>       call   cursor_shape_fix
                             <1>       ; al = 14, ah = 15  (If [CHAR_HEIGHT] = 16)
00001629 668906              <1>       mov    [esi], ax
                             <1> 
0000162C 56                  <1>       push   esi ; *
                             <1> 
0000162D 8A25[C95E0000]      <1>       mov    ah, [VGA_MODESET_CTL]
00001633 80E408              <1>       and    ah, 8 ; default palette loading ?
00001636 7524                <1>       jnz    short _sm_6
00001638 66BAC603            <1>       mov    dx, 3C6h ; VGAREG_PEL_MASK (DAC mask register)
0000163C B0FF                <1>       mov    al, 0FFh ; PEL mask
0000163E EE                  <1>       out    dx, al
0000163F 8A27                <1>       mov    ah, [edi] ; DAC model (selection number)
00001641 E8ED0F0000          <1>       call   load_dac_palette
                             <1>       ; ecx = 0
00001646 F605[C95E0000]02    <1>       test   byte [VGA_MODESET_CTL], 2 ; gray scale summing
0000164D 740D                <1>       jz     short _sm_6
0000164F 53                  <1>       push   ebx
00001650 29DB                <1>       sub    ebx, ebx ; sub bl, bl
00001652 66B90001            <1>         mov     cx, 256
```

```
00001656 E82B100000           <1>       call   gray_scale_summing
0000165B 5B                   <1>       pop    ebx
                              <1> _sm_6:
                              <1>       ; Reset Attribute Ctl flip-flop
0000165C 66BADA03             <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00001660 EC                   <1>       in     al, dx
                              <1>       ; Set Attribute Ctl
00001661 89DE                 <1>       mov    esi, ebx ; addr of params tbl for selected mode
00001663 83C623               <1>       add    esi, 35  ; actl regs
00001666 30E4                 <1>       xor    ah, ah ; 0
00001668 66BAC003             <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
                              <1> _sm_7:
0000166C 88E0                 <1>       mov    al, ah
0000166E EE                   <1>       out    dx, al ; index
0000166F AC                   <1>       lodsb
                              <1>       ; DX = 3C0h = VGAREG_ACTL_WRITE_DATA
00001670 EE                   <1>       out    dx, al ; value
00001671 FEC4                 <1>       inc    ah
00001673 80FC14               <1>       cmp    ah, 20 ; number of actl registers
00001676 72F4                 <1>       jb     short _sm_7
                              <1>       ;
00001678 88E0                 <1>       mov    al, ah ; 20
0000167A EE                   <1>       out    dx, al ; index
0000167B 28C0                 <1>       sub    al, al ; 0
0000167D EE                   <1>       out    dx, al ; value
                              <1>       ;
                              <1>       ; Set Sequencer Ctl
0000167E 89DE                 <1>       mov    esi, ebx ; addr of params tbl for selected mode
00001680 83C605               <1>       add    esi, 5 ; sequ regs
                              <1>       ;
00001683 66BAC403             <1>       mov    dx, 3C4h  ; VGAREG_SEQU_ADDRESS
00001687 EE                   <1>       out    dx, al ; 0
00001688 6642                 <1>       inc    dx ; 3C5h ; VGAREG_SEQU_DATA
0000168A B003                 <1>       mov    al, 3
0000168C EE                   <1>       out    dx, al
0000168D B401                 <1>       mov    ah, 1
                              <1> _sm_8:
0000168F 88E0                 <1>       mov    al, ah
                              <1>       ;mov   dx, 3C4h ; VGAREG_SEQU_ADDRESS
00001691 664A                 <1>       dec    dx
00001693 EE                   <1>       out    dx, al ; index
00001694 AC                   <1>       lodsb
00001695 6642                 <1>       inc    dx ; 3C5h ; VGAREG_SEQU_DATA
00001697 EE                   <1>       out    dx, al
00001698 80FC04               <1>       cmp    ah, 4 ; number of sequ regs
0000169B 7304                 <1>       jnb    short _sm_9
0000169D FEC4                 <1>       inc    ah
0000169F EBEE                 <1>       jmp    short _sm_8
                              <1> _sm_9:
                              <1>       ; Set Grafx Ctl
000016A1 89DE                 <1>       mov    esi, ebx ; addr of params tbl for selected mode
000016A3 83C637               <1>       add    esi, 55 ; grdc regs
000016A6 30E4                 <1>       xor    ah, ah ; 0
                              <1> _sm_10:
000016A8 88E0                 <1>       mov    al, ah
000016AA 66BACE03             <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
000016AE EE                   <1>       out    dx, al
000016AF AC                   <1>       lodsb
000016B0 6642                 <1>       inc    dx ; 3CFh ; VGAREG_GRDC_DATA
000016B2 EE                   <1>       out    dx, al
000016B3 FEC4                 <1>       inc    ah
000016B5 80FC09               <1>       cmp    ah, 9 ; number of grdc regs
000016B8 72EE                 <1>       jb     short _sm_10
                              <1>       ;
                              <1>       ; Disable CRTC write protection
000016BA 66BAD403             <1>       mov    dx, 3D4h  ; VGAREG_VGA_CRTC_ADDRESS
                              <1>       ;mov   al, 11h
                              <1>       ;our   dx, al
                              <1>       ;inc   dx
                              <1>       ;sub   al, al
                              <1>       ;out   dx, al
000016BE 66B81100             <1>       mov    ax, 11h
000016C2 66EF                 <1>       out    dx, ax
000016C4 89DE                 <1>       mov    esi, ebx ; addr of params tbl for selected mode
000016C6 83C60A               <1>       add    esi, 10 ; crtc regs
                              <1>       ; ah = 0
                              <1> _sm_11:
000016C9 88E0                 <1>       mov    al, ah
                              <1>       ; dx = 3D4h = VGAREG_VGA_CRTC_ADDRESS
000016CB EE                   <1>       out    dx, al ; index
000016CC AC                   <1>       lodsb
000016CD 6642                 <1>       inc    dx  ; VGAREG_VGA_CRTC_ADDRESS + 1
000016CF EE                   <1>       out    dx, al ; value
000016D0 80FC18               <1>       cmp    ah, 24 ; number of crtc registers - 1
000016D3 7306                 <1>       jnb    short _sm_12
000016D5 FEC4                 <1>       inc    ah
000016D7 664A                 <1>       dec    dx ; 3D4h
000016D9 EBEE                 <1>       jmp    short _sm_11
                              <1> _sm_12:
                              <1>       ; Set the misc register
000016DB 66BACC03             <1>       mov    dx, 3CCh ; VGAREG_READ_MISC_OUTPUT
000016DF 8A4309               <1>       mov    al, [ebx+9] ; misc reg
000016E2 EE                   <1>       out    dx, al
                              <1>       ;
                              <1>       ; Enable video
000016E3 66BAC003             <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
000016E7 B020                 <1>       mov    al, 20h
000016E9 EE                   <1>         out      dx, al   ; set bit 5 to 1
000016EA 66BADA03             <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
000016EE EC                   <1>       in     al, dx
                              <1>       ;
000016EF 803D[D3650100]00     <1>       cmp    byte [noclearmem], 0
000016F6 7740                 <1>         ja      short _sm_15
```

```
                              <1>
                              <1>       ; 29/07/2016
000016F8 31C0                 <1>       xor    eax, eax
000016FA B900400000           <1>       mov    ecx, 4000h ; 16K words (32K)
000016FF 803D[EA650100]02     <1>       cmp     byte [VGA_MTYPE], 2  ; CTEXT, MTEXT, CGA
00001706 7715                 <1>       ja     short _sm_14    ; no ? (0A0000h)
00001708 BF00800B00           <1>       mov    edi, 0B8000h
0000170D 7409                 <1>       je     short _sm_13 ; CGA graphics mode
                              <1>       ; 08/08/2016
0000170F A3[E6650100]         <1>       mov    [VGA_INT43H], eax ; 0 ; default font
00001714 66B82007             <1>       mov    ax, 0720h    ; CGA text mode
                              <1> _sm_13:
00001718 F366AB               <1>       rep    stosw
0000171B EB1B                 <1>       jmp    short _sm_15
                              <1>
                              <1> _sm_14:
0000171D BF00000A00           <1>       mov    edi, 0A0000h
                              <1>       ; ecx = 16384 dwords  (64K)
00001722 66BAC403             <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
00001726 B002                 <1>       mov    al, 2
00001728 EE                   <1>       out    dx, al
                              <1>       ;mov   dx, 3C5h ; VGAREG_SEQU_DATA
00001729 6642                 <1>       inc    dx
0000172B EC                   <1>       in     al, dx ; mmask
0000172C 6650                 <1>       push   ax
0000172E B00F                 <1>       mov    al, 0Fh ; all planes
00001730 EE                   <1>       out    dx, al
00001731 30C0                 <1>       xor    al, al ; 0
00001733 F3AB                 <1>       rep    stosd  ; ecx = 163684 (64K)
00001735 6658                 <1>       pop    ax
00001737 EE                   <1>       out    dx, al  ; mmask
                              <1> _sm_15:
                              <1>       ; ebx = addr of params tbl for selected mode
                              <1>       ; 10/08/2016
00001738 668B03               <1>       mov    ax, [ebx] ; num of columns, 'twidth'
0000173B A2[C45E0000]         <1>       mov    [CRT_COLS], al
                              <1>       ;; 26/07/2016
                              <1>       ;; CRTC_ADDRESS = 3D4h (always)
                              <1>       ;mov   ah, [ebx+1] ; num of rows, 'theightm1'
00001740 FEC4                 <1>       inc    ah ; 09/07/2016
00001742 8825[CA5E0000]       <1>       mov    [VGA_ROWS], ah
                              <1>       ; 10/08/2016
00001748 8A4302               <1>       mov    al, [ebx+2]
0000174B A2[C65E0000]         <1>       mov    [CHAR_HEIGHT], al
                              <1>       ; 29/07/2016
                              <1>       ; length of regen buffer in bytes
00001750 668B4B03             <1>       mov    cx, [ebx+3] ; 'slength_l'
00001754 66890D[D4650100]     <1>       mov    [CRT_LEN], cx
                              <1>       ;
                              <1>       ; 27/07/2016
0000175B 30E4                 <1>       xor    ah, ah
0000175D A0[66580100]         <1>       mov    al, [ACTIVE_PAGE] ; may be > 0 for mode 3
                              <1>       ;mul   word [CRT_LEN] ; 4096 for mode 3
00001762 66F7E1               <1>       mul    cx ; 29/07/2016
00001765 66A3[54580100]       <1>       mov    [CRT_START], ax
                              <1>       ;
0000176B B060                 <1>       mov    al, 60h
0000176D 803D[D3650100]00     <1>       cmp    byte [noclearmem], 0
00001774 7602                 <1>       jna    short _sm_16
00001776 0480                 <1>       add    al, 80h
                              <1> _sm_16:
00001778 A2[C75E0000]         <1>       mov    [VGA_VIDEO_CTL], al
0000177D C605[C85E0000]F9     <1>       mov    byte [VGA_SWITCHES], 0F9h
00001784 8025[C95E0000]7F     <1>       and    byte [VGA_MODESET_CTL], 7Fh
                              <1>
0000178B 5E                   <1>       pop    esi ; *
                              <1>
                              <1>       ; 26/07/2016
                              <1>       ; 07/07/2016
0000178C 668B0D[DB5E0000]     <1>       mov    cx, [CURSOR_MODE] ; restore cursor mode (initial value)
00001793 66870E               <1>       xchg   cx, [esi] ; cl = start line, ch = end line
                              <1>                      ; reset to initial value
00001796 86E9                 <1>       xchg   ch, cl   ; ch = start line, cl = end line
00001798 66890D[DB5E0000]     <1>       mov    [CURSOR_MODE], cx ; save (fixed) cursor mode
                              <1>
                              <1>       ; 27/07/2016
0000179F 803D[EA650100]02     <1>       cmp    byte [VGA_MTYPE], 2 ; CTEXT, MTEXT
000017A6 7317                 <1>       jnb    short _sm_17
                              <1>
                              <1>       ; Set cursor shape
                              <1>       ;mov   cx, 0607h
                              <1>       ;call  _set_ctype
                              <1>
                              <1>       ; 29/07/2016
000017A8 B40A                 <1>       mov    ah, 10 ; 6845 register for cursor set
000017AA E8C4050000           <1>       call   m16    ; output cx register
                              <1>
                              <1>       ; 25/07/2016
000017AF 803D[C25E0000]03     <1>         cmp     byte [CRT_MODE], 03h
000017B6 7507                 <1>       jne    short _sm_17
                              <1>       ; 26/07/2016
                              <1>
000017B8 A0[66580100]         <1>       mov    al, [ACTIVE_PAGE]
000017BD EB0C                 <1>       jmp    short _sm_18
                              <1> _sm_17:
                              <1>       ; Set cursor pos for page 0..7
000017BF 6629C0               <1>       sub    ax, ax ; eax = 0
000017C2 BF[56580100]         <1>       mov    edi, CURSOR_POSN
000017C7 AB                   <1>       stosd
000017C8 AB                   <1>       stosd
000017C9 AB                   <1>       stosd
000017CA AB                   <1>       stosd
                              <1>       ;; Set active page 0
```

```
                             <1>        ;mov   [ACTIVE_PAGE], al ; 0
                             <1> _sm_18:
                             <1>        ; 29/07/2016
000017CB 803D[EA650100]02    <1>        cmp    byte [VGA_MTYPE], 2 ; CTEXT, MTEXT
000017D2 0F8386000000        <1>          jnb _sm_23
                             <1>
                             <1>        ;cmp   byte [CHAR_HEIGHT], 16
                             <1>        ;je    short _sm_19
                             <1>
                             <1>        ;; copy and activate 8x16 font
                             <1>
                             <1>        ; 26/07/2016
000017D8 B004                <1>        mov    al, 04h
                             <1>        ;sub   bl, bl
                             <1>        ; AX = 1104H ; Load ROM 8x16 Character Set
                             <1>        ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
000017DA E83A150000          <1>        call   load_text_8_16_pat
                             <1>
                             <1>        ; video_func_1103h:
                             <1>        ; biosfn_set_text_block_specifier:
                             <1>        ; BL = font block selector code
                             <1>        ; NOTE: TRDOS 386 only uses and sets font block 0
                             <1>        ; (It is as BL = 0 for TRDOS 386)
000017DF 66BAC403            <1>        mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
                             <1>        ;mov   ah, bl
000017E3 28E4                <1>        sub    ah, ah ; 0
000017E5 B003                <1>        mov    al, 03h
000017E7 66EF                <1>        out    dx, ax
                             <1> _sm_19:
                             <1>        ; 29/07/2016
                             <1>        ; 26/07/2016
                             <1>        ; 24/06/2016
                             <1>        ;mov   edi, 0B8000h
                             <1>        ;mov   cx, 4000h ; 16K words (32K)
                             <1>        ;
000017E9 30C0                <1>        xor    al, al
000017EB 3805[D1650100]      <1>          cmp     byte [p_crt_mode], al ; 0
000017F1 7707                <1>          ja      short _sm_20 ; 3h, 80h or 83h
                             <1>
                             <1>        ; 30/07/2016
                             <1>        ; 24/06/2016
                             <1>        ; TRDOS 386 (TRDOS v2) 'set mode' modification
                             <1>        ; (for multiscreen feature):
                             <1>        ; If '_set_mode' procedure is called for video mode 3
                             <1>        ;     while video mode is 3, video page will be cleared
                             <1>        ;     and cursor position of video page will be reset.
                             <1>        ; If '_set_mode' procedure is called for a video mode
                             <1>        ;     except video mode 3, while current video mode
                             <1>        ;     is 3; all video pages of mode 3 will be copied
                             <1>        ;     to 98000h address as backup, before mode change.
                             <1>        ; If '_set_mode' procedure is called for video mode 3
                             <1>        ;     while video mode is not 3 and if there is video
                             <1>        ;     page backup for video mode 3, all (of 8) mode 3
                             <1>        ;     video pages will be restored from 98000h.
                             <1>
000017F3 A2[66580100]        <1>        mov    [ACTIVE_PAGE], al ; 0
                             <1>        ;mov   ax, 0720h
                             <1>        ;;mov  cx, 4000h ; 16K words (32K)
                             <1>        ;;mov  edi, 0B8000h
                             <1>        ;rep   stosw
                             <1>        ;sub   al, al
000017F8 EB64                <1>        jmp    short _sm_23
                             <1> _sm_20:
                             <1>        ; Previous video mode is 3
                             <1>        ; New video mode is 3 while current video mode is not 3
                             <1>        ; (multi screen) video pages will be restored from 0B0000h
                             <1>
000017FA 0FB61D[66580100]    <1>        movzx  ebx, byte [ACTIVE_PAGE]
00001801 D0E3                <1>        shl    bl, 1 ; * 2
00001803 81C3[56580100]      <1>        add    ebx, CURSOR_POSN
                             <1>
                             <1>        ; 29/07/2016
00001809 F605[D1650100]7F    <1>        test    byte [p_crt_mode], 7Fh ; 83h or 3h
00001810 7427                <1>        jz     short _sm_21 ; do not restore video pages
                             <1>
                             <1>        ;; restore video pages
00001812 BE00800900          <1>        mov    esi, 98000h ; 30/07/2016
00001817 BF00800B00          <1>        mov    edi, 0B8000h
0000181C 66B90020            <1>        mov    cx, 2000h ; 8K dwords (32K)
00001820 F3A5                <1>        rep    movsd
                             <1>
                             <1>        ; restore cursor positions
00001822 BE[D6650100]        <1>        mov    esi, cursor_pposn
00001827 BF[56580100]        <1>        mov    edi, CURSOR_POSN
                             <1>        ;mov   ecx, 4 ; restore all cursor positions (16 bytes)
0000182C B104                <1>        mov    cl, 4
0000182E F3A5                <1>        rep    movsd
                             <1>
00001830 F605[D1650100]80    <1>          test    byte [p_crt_mode], 80h
00001837 7420                <1>        jz     short _sm_22 ; do not clear current video pages
                             <1> _sm_21:
                             <1>        ; clear video page
00001839 668B0D[D4650100]    <1>        mov    cx, [CRT_LEN] ; 4096
00001840 66D1E9              <1>        shr    cx, 1 ; 2072
00001843 66B82007            <1>        mov    ax, 0720h
00001847 BF00800B00          <1>        mov    edi, 0B8000h ; [crt_base]
0000184C 66033D[54580100]    <1>        add    di, [CRT_START]
00001853 F366AB              <1>        rep    stosw  ; FILL THE REGEN BUFFER WITH BLANKS
                             <1>        ;
00001856 66890B              <1>        mov    [ebx], cx ; reset cursor position
                             <1> _sm_22:
00001859 A2[D1650100]        <1>        mov    [p_crt_mode], al ; 0
                             <1> _sm_23:
```

```
                             <1>       ; al = video page number
                             <1>       ; [CRT_LEN] = length of regen buffer in bytes
0000185E E81E010000          <1>       call   _set_active_page
                             <1> 
                             <1> ;-----       NORMAL RETURN FROM ALL VIDEO RETURNS
00001863 C3                  <1>       retn
                             <1> 
                             <1> cursor_shape_fix:
                             <1>       ; 07/07/2016
                             <1>       ; (Cursor start and cursor end line values -6,7-
                             <1>       ; will be fixed depending on character height)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', ' biosfn_set_cursor_shape (CH,CL)'
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      AL = cursor start line (=6)
                             <1>       ;      AH = cursor end line (=7)
                             <1>       ; OUTPUT ->
                             <1>       ;      AL = cursor start line (=14)
                             <1>       ;      AH = cursor end line (=15)
                             <1>       ;
                             <1>       ;; if((modeset_ctl&0x01)&&(cheight>8)&&(CL<8)&&(CH<0x20))
                             <1> 
                             <1>       ;test  byte [VGA_MODESET_CTL], 1 ; VGA active
                             <1>       ;jz    short csf_3
00001864 803D[C65E0000]08    <1>       cmp    byte [CHAR_HEIGHT], 8
0000186B 7649                <1>       jna    short csf_3
0000186D 80FC08              <1>       cmp    ah, 8
00001870 7344                <1>       jnb    short csf_3
00001872 3C20                <1>       cmp    al, 20h
00001874 7340                <1>       jnb    short csf_3
                             <1>       ;
00001876 6650                <1>       push   ax
                             <1>       ; {
                             <1>       ; if(CL!=(CH+1))
00001878 FEC0                <1>       inc    al
0000187A 38C4                <1>       cmp    ah, al   ; ah != al + 1
0000187C 740F                <1>         je      short csf_1
                             <1>       ; CH = ((CH+1) * cheight / 8) -1;
0000187E 8A25[C65E0000]      <1>       mov    ah, [CHAR_HEIGHT]
00001884 F6E4                <1>       mul    ah
00001886 C0E803              <1>       shr    al, 3 ; / 8
00001889 FEC8                <1>       dec    al ; - 1
0000188B EB0E                <1>       jmp    short csf_2
                             <1> csf_1:
                             <1>       ; }
                             <1>       ; else       ; ah = al + 1
                             <1>       ; {
0000188D FEC4                <1>       inc    ah     ; ah = ah + 1
                             <1>       ; CH = ((CL+1) * cheight / 8) - 2;
0000188F A0[C65E0000]        <1>       mov    al, [CHAR_HEIGHT]
00001894 F6E4                <1>       mul    ah
00001896 C0E803              <1>       shr    al, 3 ; / 8
00001899 2C02                <1>       sub    al, 2 ; - 2
                             <1>       ; al = 14 (if [CHAR_HEIGHT] = 16)
                             <1> csf_2:
0000189B 880424              <1>       mov    [esp], al
0000189E 8A642401            <1>       mov    ah, [esp+1]
                             <1>       ; CL = ((CL+1) * cheight / 8) - 1;
000018A2 FEC4                <1>       inc    ah
000018A4 A0[C65E0000]        <1>       mov    al, [CHAR_HEIGHT]
000018A9 F6E4                <1>       mul    ah
000018AB C0E803              <1>       shr    al, 3 ; / 8
000018AE FEC8                <1>       dec    al ; - 1
000018B0 88442401            <1>       mov    [esp+1], al
                             <1>       ; ah = 15 (if [CHAR_HEIGHT] = 16)
                             <1>       ;
000018B4 6658                <1>       pop    ax
                             <1> csf_3:
000018B6 C3                  <1>       retn
                             <1> 
                             <1> SET_CTYPE:
                             <1>       ; 12/09/2016
                             <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
000018B7 803D[C25E0000]07    <1>       cmp    byte [CRT_MODE], 7
000018BE 0F8790FCFFFF        <1>       ja     VIDEO_RETURN ; 12/09/2016
000018C4 E805000000          <1>       call   _set_ctype
000018C9 E986FCFFFF          <1>         jmp     VIDEO_RETURN
                             <1> 
                             <1> _set_ctype:
                             <1>       ; 02/09/2014 (Retro UNIX 386 v1)
                             <1>       ;
                             <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1> 
                             <1>       ; (CH) = BITS 4-0 = START LINE FOR CURSOR
                             <1>       ;  ** HARDWARE WILL ALWAYS CAUSE BLINK
                             <1>       ;  ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
                             <1>       ;     OR NO CURSOR AT ALL
                             <1>       ; (CL) = BITS 4-0 = END LINE FOR CURSOR
                             <1> 
                             <1> ;------------------------------------------------
                             <1> ; SET_CTYPE
                             <1> ;     THIS ROUTINE SETS THE CURSOR VALUE
                             <1> ; INPUT
                             <1> ;     (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
                             <1> ; OUTPUT
                             <1> ;     NONE
                             <1> ;------------------------------------------------
                             <1> 
                             <1>       ; 07/07/2016
```

```
                                <1>       ; Fixing cursor start and stop line depending on
                                <1>       ; current character height (=16)
                                <1>       ; (Note: Default/initial values are 6 and 7.
                                <1>       ; If set values are 6 (start) & 7 (stop) and
                                <1>       ; [CHAR_HEIGHT] = 16 :
                                <1>       ; After fixing, start line will be 14, stop line
                                <1>       ; will be 15.)
000018CE 6689C8                 <1>       mov    ax, cx
000018D1 86C4                   <1>       xchg   al, ah
                                <1>       ; AL = start line, AH = stop line
000018D3 E88CFFFFFF             <1>       call   cursor_shape_fix
                                <1>       ; AL = start line (fixed), AH = stop line (fixed)
000018D8 6689C1                 <1>       mov    cx, ax
000018DB 86E9                   <1>       xchg   ch, cl
                                <1>       ; CH = start line (fixed), CL = stop line (fixed)
                                <1>       ;
000018DD B40A                   <1>       mov    ah, 10 ; 6845 register for cursor set
000018DF 66890D[DB5E0000]       <1>       mov    [CURSOR_MODE], cx ; save in data area
                                <1>       ;call  m16    ; output cx register
                                <1>       ;retn
000018E6 E988040000             <1>         jmp     m16
                                <1>
                                <1> SET_CPOS:
                                <1>       ; 12/09/2016
                                <1>       ; 07/07/2016
                                <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
000018EB 80FF07                 <1>       cmp    bh, 7 ; video page > 7 ; 07/07/2016
000018EE 0F8760FCFFFF           <1>       ja     VIDEO_RETURN
                                <1>       ;
000018F4 803D[C25E0000]07       <1>       cmp    byte [CRT_MODE], 7
000018FB 770A                   <1>       ja     short vga_set_cpos ; 12/09/2016
000018FD E846040000             <1>       call   _set_cpos
00001902 E94DFCFFFF             <1>         jmp     VIDEO_RETURN
                                <1>
                                <1> vga_set_cpos:
                                <1>       ; 12/09/2016
                                <1>       ; 09/07/2016
                                <1>       ; set cursor position
                                <1>       ; NOTE: Hardware cursor position will not be set
                                <1>       ;   in any VGA modes (>7)
                                <1>       ;   But, cursor position will be saved into
                                <1>       ;   [CURSOR_POSN].
                                <1>       ;   TRDOS 386 (TRDOS v2.0) uses only one page
                                <1>       ;   (page 0) for all graphics modes.
                                <1>
00001907 668915[56580100]       <1>       mov    [CURSOR_POSN], dx ; save cursor pos for pg 0
                                <1>       ; 04/08/2016
                                <1>       ;mov   bh, [ACTIVE_PAGE] ; = 0
                                <1>       ;call _set_cpos
0000190E E941FCFFFF             <1>       jmp     VIDEO_RETURN
                                <1>
                                <1> READ_CURSOR:
                                <1>       ; 12/09/2016
                                <1>       ; 07/07/2016
                                <1>       ; 12/05/2016
                                <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                                <1>       ;
                                <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                                <1>
                                <1> ;-----------------------------------------------------
                                <1> ; READ_CURSOR
                                <1> ;     THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
                                <1> ;     845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
                                <1> ; INPUT
                                <1> ;     BH - PAGE OF CURSOR
                                <1> ; OUTPUT
                                <1> ;     DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
                                <1> ;     CX - CURRENT CURSOR MODE
                                <1> ;-----------------------------------------------------
                                <1>
                                <1>       ; BH = Video page number (0 to 7)
                                <1>
                                <1>       ; 07/07/2016
00001913 80FF07                 <1>       cmp    bh, 7 ; video page > 7 (invalid)
00001916 7606                   <1>       jna    short read_cursor_1
                                <1>       ; invalid video page (input)
00001918 31C9                   <1>       xor    ecx, ecx ; 0
0000191A 31D2                   <1>       xor    edx, edx ; 0
0000191C EB15                   <1>       jmp    short read_cursor_2
                                <1> read_cursor_1:
                                <1>       ; 12/09/2016
0000191E 803D[C25E0000]07       <1>       cmp    byte [CRT_MODE], 7 ; vga mode
00001925 7727                   <1>       ja     short vga_get_cpos
                                <1>       ;
00001927 E815000000             <1>       call   get_cpos
0000192C 0FB70D[DB5E0000]       <1>       movzx  ecx, word [CURSOR_MODE]
                                <1> read_cursor_2:
00001933 5D                     <1>       pop    ebp
00001934 5F                     <1>       pop    edi
00001935 5E                     <1>       pop    esi
00001936 5B                     <1>       pop    ebx
00001937 58                     <1>       pop    eax  ; DISCARD SAVED CX AND DX
00001938 58                     <1>       pop    eax
00001939 A1[C4650100]           <1>       mov    eax, [video_eax] ; 12/05/2016
                                <1>       ;;15/01/2017
                                <1>       ;;mov  byte [intflg], 0 ; 07/01/2017
0000193E 1F                     <1>       pop    ds
0000193F 07                     <1>       pop    es
00001940 CF                     <1>       iretd
                                <1>
                                <1> get_cpos:
                                <1>       ; 12/05/2016
                                <1>       ; 16/01/2016
```

```
                             <1>        ; BH = Video page number (0 to 7)
                             <1>        ;
00001941 D0E7                <1>        shl    bh, 1 ; WORD OFFSET
00001943 0FB6F7              <1>        movzx  esi, bh
00001946 0FB796[56580100]    <1>        movzx  edx, word [esi+CURSOR_POSN]
0000194D C3                  <1>        retn
                             <1> 
                             <1> vga_get_cpos:
                             <1>        ; 12/09/2016
                             <1>        ; get cursor position (vga)
0000194E 0FB715[56580100]    <1>        movzx  edx, word [CURSOR_POSN] ; cursor pos for pg 0
00001955 31C9                <1>        xor    ecx, ecx ; Cursor Mode = 0 (invalid)
00001957 EBDA                <1>        jmp     short read_cursor_2
                             <1> 
                             <1> ACT_DISP_PAGE:
                             <1>        ; 07/07/2016
                             <1>        ; 26/06/2016
                             <1>        ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>        ;
                             <1>        ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1>        ;
                             <1> ;-----------------------------------------------------
                             <1> ; ACT_DISP_PAGE
                             <1> ;     THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
                             <1> ;     THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT
                             <1> ; INPUT
                             <1> ;     AL HAS THE NEW ACTIVE DISPLAY PAGE
                             <1> ; OUTPUT
                             <1> ;     THE 6845 IS RESET TO DISPLAY THAT PAGE
                             <1> ;-----------------------------------------------------
                             <1>        ; 07/07/2016
00001959 3C07                <1>        cmp    al, 7  ; > 7 = invalid video page number
0000195B 0F87F3FBFFFF        <1>        ja     VIDEO_RETURN
00001961 803D[C25E0000]03    <1>          cmp     byte [CRT_MODE], 3
00001968 7408                <1>        je     short adp_1
0000196A 20C0                <1>        and    al, al
0000196C 0F85E2FBFFFF        <1>          jnz     VIDEO_RETURN
                             <1>        ;sub   al, al ; 0 ; force to page 0
                             <1> adp_1:
00001972 E805000000          <1>        call   set_active_page
00001977 E9D8FBFFFF          <1>          jmp     VIDEO_RETURN
                             <1> 
                             <1> set_active_page:   ; tty_sw
                             <1>        ; 09/12/2017
                             <1>        ; 26/07/2016
                             <1>        ; 26/06/2016
                             <1>        ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>        ; 30/06/2015
                             <1>        ; 04/03/2014  (act_disp_page --> tty_sw)
                             <1>        ; 10/12/2013
                             <1>        ; 04/12/2013
                             <1>        ;
0000197C A2[66580100]        <1>        mov    [ACTIVE_PAGE], al ; save active page value ; [ptty]
                             <1> _set_active_page:
                             <1>        ; 27/06/2015
00001981 0FB6D8              <1>        movzx  ebx, al
                             <1>        ;
                             <1>        ;cbw   ; 07/09/2014 (ah=0)
00001984 28E4                <1>        sub    ah, ah ; 09/12/2017
00001986 66F725[D4650100]    <1>        mul    word [CRT_LEN]  ; get saved length of regen buffer
                             <1>                            ; display page times regen length
                             <1>        ; 10/12/2013
0000198D 66A3[54580100]      <1>        mov    [CRT_START], ax ; save start address for later
00001993 6689C1              <1>        mov    cx, ax ; start address to cx
                             <1> _M16:
                             <1>        ;sar   cx, 1
00001996 66D1E9              <1>        shr    cx, 1  ; divide by 2 for 6845 handling
00001999 B40C                <1>        mov    ah, 12 ; 6845 register for start address
0000199B E8D3030000          <1>        call   m16
                             <1>        ;sal   bx, 1
                             <1>        ; 01/09/2014
000019A0 D0E3                <1>        shl    bl, 1  ; *2 for word offset
000019A2 81C3[56580100]      <1>        add    ebx, CURSOR_POSN
000019A8 668B13              <1>        mov    dx, [ebx] ; get cursor for this page
                             <1>        ; 16/01/2016
                             <1>        ;call  m18
                             <1>        ;retn
000019AB E9AF030000          <1>        jmp    m18
                             <1> 
                             <1> position:
                             <1>        ; 24/06/2016
                             <1>        ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>        ; 27/06/2015
                             <1>        ; 02/09/2014
                             <1>        ; 30/08/2014 (Retro UNIX 386 v1)
                             <1>        ; 04/12/2013 (Retro UNIX 8086 v1)
                             <1>        ;
                             <1>        ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1>        ;
                             <1> ;----------------------------------------
                             <1> ; POSITION
                             <1> ;     THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
                             <1> ;     OF A CHARACTER IN THE ALPHA MODE
                             <1> ; INPUT
                             <1> ;     AX = ROW, COLUMN POSITION
                             <1> ; OUTPUT
                             <1> ;     AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
                             <1> ;----------------------------------------
                             <1> 
                             <1>        ; DX = ROW, COLUMN POSITION
000019B0 0FB605[C45E0000]    <1>        movzx  eax, byte [CRT_COLS] ; 24/06/2016
000019B7 F6E6                <1>        mul    dh      ; row value
000019B9 30F6                <1>        xor    dh, dh   ; 0
```

```
000019BB 6601D0                  <1>       add    ax, dx  ; add column value to the result
000019BE 66D1E0                  <1>       shl    ax, 1  ; * 2 for attribute bytes
                                 <1>              ; EAX = AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
000019C1 C3                      <1>       retn
                                 <1>
                                 <1> find_position:
                                 <1>       ; 24/06/2016
                                 <1>       ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
                                 <1>       ; 27/06/2015
                                 <1>       ; 07/09/2014
                                 <1>       ; 02/09/2014
                                 <1>       ; 30/08/2014 (Retro UNIX 386 v1)
                                 <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                                 <1>
000019C2 0FB6CF                  <1>       movzx  ecx, bh ; video page number
000019C5 89CE                    <1>       mov    esi, ecx
000019C7 66D1E6                  <1>       shl    si, 1
000019CA 668B96[56580100]        <1>       mov    dx, [esi+CURSOR_POSN]
000019D1 740C                    <1>       jz     short p21
000019D3 6631F6                  <1>       xor    si, si
                                 <1> p20:
000019D6 660335[D4650100]        <1>       add    si, [CRT_LEN] ; 24/06/2016
                                 <1>       ;add   si, 80*25*2 ; add length of buffer for one page
000019DD E2F7                    <1>       loop   p20
                                 <1> p21:
000019DF 6621D2                  <1>       and    dx, dx
000019E2 7407                    <1>       jz     short p22
000019E4 E8C7FFFFFF              <1>       call   position ; determine location in regen in page
000019E9 01C6                    <1>       add    esi, eax ; add location to start of regen page
                                 <1> p22:
                                 <1>       ;mov   dx, [addr_6845] ; get base address of active display
                                 <1>       ;mov   dx, 03D4h ; I/O address of color card
                                 <1>       ;add   dx, 6  ; point at status port
000019EB 66BADA03                <1>       mov    dx, 03DAh ; status port
                                 <1>       ; cx = 0
000019EF C3                      <1>       retn
                                 <1>
                                 <1> SCROLL_UP:
                                 <1>       ; 07/07/2016
                                 <1>       ; 26/06/2016
                                 <1>       ; 12/05/2016
                                 <1>       ; 30/01/2016
                                 <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                                 <1>       ; 07/09/2014
                                 <1>       ; 02/09/2014
                                 <1>       ; 01/09/2014 (Retro UNIX 386 v1 - beginning)
                                 <1>       ; 04/04/2014
                                 <1>       ; 04/12/2013
                                 <1>       ;
                                 <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                                 <1>       ;
                                 <1> ;----------------------------------------------
                                 <1> ; SCROLL UP
                                 <1> ;     THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
                                 <1> ;     ON THE SCREEN
                                 <1> ; INPUT
                                 <1> ;     (AH) = CURRENT CRT MODE
                                 <1> ;     (AL) = NUMBER OF ROWS TO SCROLL
                                 <1> ;     (CX) = ROW/COLUMN OF UPPER LEFT CORNER
                                 <1> ;     (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
                                 <1> ;     (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
                                 <1> ;     (DS) = DATA SEGMENT
                                 <1> ;     (ES) = REGEN BUFFER SEGMENT
                                 <1> ; OUTPUT
                                 <1> ;     NONE -- THE REGEN BUFFER IS MODIFIED
                                 <1> ;--------------------------------------------
                                 <1>
                                 <1>       ; 07/07/2016
000019F0 38F5                    <1>       cmp    ch, dh
000019F2 0F875CFBFFFF            <1>       ja     VIDEO_RETURN
000019F8 38D1                    <1>       cmp    cl, dl
000019FA 0F8754FBFFFF            <1>       ja     VIDEO_RETURN
                                 <1>       ;
00001A00 E805000000              <1>       call   _scroll_up
00001A05 E94AFBFFFF              <1>         jmp     VIDEO_RETURN
                                 <1>
                                 <1> _scroll_up:  ; from 'write_tty'
                                 <1>       ;
                                 <1>       ; cl = left upper column
                                 <1>       ; ch = left upper row
                                 <1>       ; dl = right lower column
                                 <1>       ; dh = right lower row
                                 <1>       ;
                                 <1>       ; al = line count
                                 <1>       ; bl = attribute to be used on blanked line
                                 <1>       ; bh = video page number (0 to 7)
                                 <1>
00001A0A E896000000              <1>       call   test_line_count ; 16/01/2016
                                 <1>
00001A0F 8A25[C25E0000]          <1>       mov    ah, [CRT_MODE] ; current video mode
                                 <1>       ;cmp   ah, 4
                                 <1>       ;jb    short n0
                                 <1>       ;cmp   byte [CRT_MODE], 4
00001A15 80FC04                  <1>         cmp  ah, 4 ; 07/07/2016
00001A18 0F8320050000            <1>       jnb     GRAPHICS_UP ; 26/06/2016
                                 <1>
                                 <1>       ;cmp   ah, 7 ; TEST FOR BW CARD
                                 <1>       ;jne   GRAPHICS_UP
                                 <1> n0:
                                 <1>       ; 07/07/2016
00001A1E 80FF07                  <1>       cmp    bh, 7 ; video page number
00001A21 7606                    <1>       jna    short n1
00001A23 8A3D[66580100]          <1>       mov    bh, [ACTIVE_PAGE]
```

```
                                  <1> n1:
00001A29 88DC                     <1>        mov    ah, bl ; attribute
00001A2B 6650                     <1>        push   ax ; *
                                  <1>        ;mov   esi, [CRT_BASE]
00001A2D BE00800B00               <1>          mov     esi, 0B8000h
00001A32 3A3D[66580100]           <1>          cmp     bh, [ACTIVE_PAGE]
00001A38 750B                     <1>        jne    short n2
                                  <1>        ;
00001A3A 66A1[54580100]           <1>          mov     ax, [CRT_START]
00001A40 6601C6                   <1>          add     si, ax
00001A43 EB11                     <1>          jmp     short n4
                                  <1> n2:
00001A45 20FF                     <1>          and     bh, bh
00001A47 740D                     <1>        jz     short n4
00001A49 88F8                     <1>        mov    al, bh
                                  <1> n3:
00001A4B 660335[D4650100]         <1>          add  si, [CRT_LEN]
00001A52 FEC8                     <1>          dec  al
00001A54 75F5                     <1>        jnz    short n3
                                  <1> n4:
00001A56 E85D000000               <1>        call   scroll_position ; 16/01/2016
00001A5B 7420                     <1>          jz      short n6
                                  <1>
00001A5D 01CE                     <1>          add     esi, ecx ; from address for scroll
00001A5F 88F5                     <1>        mov    ch, dh  ; #rows in block
00001A61 28C5                     <1>        sub    ch, al ; #rows to be moved
                                  <1> n5:
00001A63 E894000000               <1>        call   n10 ; 16/01/2016
                                  <1>
00001A68 51                       <1>          push ecx
00001A69 0FB60D[C45E0000]         <1>        movzx  ecx, byte [CRT_COLS]
00001A70 00C9                     <1>        add    cl, cl
00001A72 01CE                     <1>          add  esi, ecx  ; next line
00001A74 01CF                     <1>          add  edi, ecx
00001A76 59                       <1>        pop    ecx
                                  <1>
00001A77 FECD                     <1>        dec    ch      ; count of lines to move
00001A79 75E8                     <1>        jnz    short n5 ; row loop
                                  <1>        ; ch = 0
00001A7B 88C6                     <1>        mov    dh, al  ; #rows
                                  <1> n6:
                                  <1>        ; attribute in ah
00001A7D B020                     <1>        mov    al, ' '        ; fill with blanks
                                  <1> n7:
00001A7F E885000000               <1>        call   n11 ; 16/01/2016
                                  <1>
00001A84 8A0D[C45E0000]           <1>        mov    cl, [CRT_COLS]
00001A8A 00C9                     <1>        add    cl, cl
00001A8C 01CF                     <1>          add  edi, ecx
                                  <1>
00001A8E FECE                     <1>        dec    dh
00001A90 75ED                     <1>        jnz    short n7
                                  <1> n16:
00001A92 3A3D[66580100]           <1>        cmp    bh, [ACTIVE_PAGE]
00001A98 750A                     <1>        jne    short n8
                                  <1>
                                  <1>        ;cmp   byte [CRT_MODE], 7  ; is this the black and white card
                                  <1>        ;je    short n8            ; if so, skip the mode reset
                                  <1>
00001A9A A0[C35E0000]             <1>        mov    al, [CRT_MODE_SET] ; get the value of mode set
00001A9F 66BAD803                 <1>        mov    dx, 03D8h ; always set color card port
00001AA3 EE                       <1>        out    dx, al
                                  <1> n8:
00001AA4 C3                       <1>        retn
                                  <1>
                                  <1> test_line_count:
                                  <1>        ; 12/05/2016
                                  <1>        ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                                  <1>        ; 07/09/2014 (scroll_up)
00001AA5 08C0                     <1>        or     al, al
00001AA7 740E                     <1>        jz     short al_set2
00001AA9 6652                     <1>        push   dx
00001AAB 28EE                     <1>        sub    dh, ch  ; subtract upper row from lower row number
00001AAD FEC6                     <1>        inc    dh     ; adjust difference by 1
00001AAF 38C6                     <1>        cmp    dh, al       ; line count = amount of rows in window?
00001AB1 7502                     <1>        jne    short al_set1 ; if not the we're all set
00001AB3 30C0                     <1>        xor    al, al ; otherwise set al to zero
                                  <1> al_set1:
00001AB5 665A                     <1>        pop    dx
                                  <1> al_set2:
00001AB7 C3                       <1>        retn
                                  <1>
                                  <1> scroll_position:
                                  <1>        ; 26/06/2016
                                  <1>        ; 30/01/2016
                                  <1>          ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                                  <1>        ; 07/09/2014 (scroll_up)
                                  <1>
00001AB8 6652                     <1>        push   dx
00001ABA 6689CA                   <1>        mov    dx, cx ; now, upper left position in DX
00001ABD E8EEFEFFFF               <1>        call   position
00001AC2 01C6                     <1>        add    esi, eax
00001AC4 89F7                     <1>        mov    edi, esi
00001AC6 665A                     <1>        pop    dx     ; lower right position in DX
00001AC8 6629CA                   <1>        sub    dx, cx
00001ACB FEC6                     <1>        inc    dh     ; dh = #rows
00001ACD FEC2                     <1>        inc    dl     ; dl = #cols in block
00001ACF 59                       <1>        pop    ecx    ; return address
00001AD0 6658                     <1>        pop    ax     ; * ; al = line count, ah = attribute
00001AD2 51                       <1>        push   ecx    ; return address
00001AD3 0FB7C8                   <1>        movzx  ecx, ax
00001AD6 8A25[C45E0000]           <1>        mov    ah, [CRT_COLS]
00001ADC F6E4                     <1>        mul    ah     ; determine offset to from address
```

```
00001ADE 6601C0              <1>       add    ax, ax  ; *2 for attribute byte
                             <1>       ;
00001AE1 6650                <1>       push   ax     ; offset
00001AE3 6652                <1>       push   dx
                             <1>       ;
                             <1>       ; 04/04/2014
00001AE5 66BADA03            <1>       mov    dx, 3DAh ; guaranteed to be color card here
                             <1> n9:                      ; wait_display_enable
00001AE9 EC                  <1>         in      al, dx   ; get port
00001AEA A808                <1>       test   al, RVRT ; wait for vertical retrace
00001AEC 74FB                <1>       jz     short n9 ; wait_display_enable
00001AEE B025                <1>       mov    al, 25h
00001AF0 B2D8                <1>       mov    dl, 0D8h ; address control port
00001AF2 EE                  <1>       out    dx, al ; turn off video during vertical retrace
00001AF3 665A                <1>       pop    dx     ; #rows, #cols
00001AF5 6658                <1>             pop    ax     ; offset
00001AF7 6691                <1>       xchg   ax, cx ;
                             <1>       ; ecx = offset, al = line count, ah = attribute
                             <1>       ;
00001AF9 08C0                <1>       or     al, al
00001AFB C3                  <1>       retn
                             <1> n10:
                             <1>       ; Move rows
00001AFC 88D1                <1>       mov    cl, dl ; get # of cols to move
00001AFE 56                  <1>       push   esi
00001AFF 57                  <1>       push   edi    ; save start address
                             <1> n10r:
00001B00 66A5                <1>       movsw        ; move that line on screen
00001B02 FEC9                <1>       dec    cl
00001B04 75FA                <1>         jnz     short n10r
00001B06 5F                  <1>       pop    edi
00001B07 5E                  <1>       pop    esi    ; recover addresses
00001B08 C3                  <1>       retn
                             <1> n11:
                             <1>       ; Clear rows
                             <1>                    ; dh =  #rows
00001B09 88D1                <1>         mov  cl, dl ; get # of cols to clear
00001B0B 57                  <1>         push    edi     ; save address
                             <1> n11r:
00001B0C 66AB                <1>         stosw           ; store fill character
00001B0E FEC9                <1>       dec    cl
00001B10 75FA                <1>         jnz     short n11r
00001B12 5F                  <1>         pop     edi     ; recover address
00001B13 C3                  <1>       retn
                             <1>
                             <1> SCROLL_DOWN:
                             <1>       ; 07/07/2016
                             <1>       ; 27/06/2016
                             <1>       ; 26/06/2016
                             <1>       ; 12/05/2016
                             <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1>
                             <1> ;-------------------------------------------
                             <1> ; SCROLL DOWN
                             <1> ;     THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
                             <1> ;     BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
                             <1> ;     WITH A DEFINED CHARACTER
                             <1> ; INPUT
                             <1> ;     (AH) = CURRENT CRT MODE
                             <1> ;     (AL) = NUMBER OF LINES TO SCROLL
                             <1> ;     (CX) = UPPER LEFT CORNER OF RECION
                             <1> ;     (DX) = LOWER RIGHT CORNER OF REGION
                             <1> ;     (BH) = FILL CHARACTER
                             <1> ;     (DS) = DATA SEGMENT
                             <1> ;     (ES) = REGEN SEGMENT
                             <1> ; OUTPUT
                             <1> ;     NONE -- SCREEN IS SCROLLED
                             <1> ;-------------------------------------------
                             <1>
                             <1>       ; 07/07/2016
00001B14 38F5                <1>       cmp    ch, dh
00001B16 0F8738FAFFFF        <1>       ja     VIDEO_RETURN
00001B1C 38D1                <1>       cmp    cl, dl
00001B1E 0F8730FAFFFF        <1>       ja     VIDEO_RETURN
                             <1>       ;
00001B24 E805000000          <1>       call   _scroll_down
00001B29 E926FAFFFF          <1>         jmp     VIDEO_RETURN
                             <1>
                             <1> _scroll_down: ; 27/06/2016
                             <1>
                             <1>       ; cl = left upper column
                             <1>       ; ch = left upper row
                             <1>       ; dl = right lower column
                             <1>       ; dh = right lower row
                             <1>       ;
                             <1>       ; al = line count
                             <1>       ; bl = attribute to be used on blanked line
                             <1>       ; bh = video page number (0 to 7)
                             <1>
                             <1>       ; !!!!
00001B2E FD                  <1>       std          ; DIRECTION FOR SCROLL DOWN
                             <1>       ; !!!!
00001B2F E871FFFFFF          <1>       call   test_line_count ; 16/01/2016
                             <1>
00001B34 8A25[C25E0000]      <1>       mov    ah, [CRT_MODE] ; current video mode
                             <1>       ;cmp   ah, 4
                             <1>       ;jb    short _n0
                             <1>       ;cmp   byte [CRT_MODE], 4
00001B3A 80FC04              <1>         cmp ah, 4 ; 07/07/2016
00001B3D 0F83DF070000        <1>       jnb     GRAPHICS_DOWN ; 26/06/2016
                             <1>
                             <1>       ;cmp   ah, 7 ; TEST FOR BW CARD
                             <1>       ;jne   GRAPHICS_DOWN
                             <1> _n0:
                             <1>       ; 07/07/2016
00001B43 80FF07              <1>       cmp    bh, 7 ; video page number
00001B46 7606                <1>       jna    short n12
00001B48 8A3D[66580100]      <1>       mov    bh, [ACTIVE_PAGE]
                             <1>       ;
                             <1> n12:               ; CONTINUE_DOWN
00001B4E 88DC                <1>       mov    ah, bl
00001B50 6650                <1>       push   ax     ; * ; save attribute in ah
00001B52 6689D0              <1>       mov    ax, dx ; LOWER RIGHT CORNER
00001B55 E85EFFFFFF          <1>       call   scroll_position     ; GET REGEN LOCATION
00001B5A 741F                <1>       jz     short n14
00001B5C 29CE                <1>       sub    esi, ecx  ; SI IS FROM ADDRESS
00001B5E 88F5                <1>       mov    ch, dh   ; #rows in block
00001B60 28C5                <1>       sub    ch, al ; #rows to be moved
                             <1> n13:
00001B62 E895FFFFFF          <1>       call   n10    ; MOVE ONE ROW
                             <1>
00001B67 51                  <1>       push   ecx
00001B68 8A0D[C45E0000]      <1>       mov    cl, [CRT_COLS]
00001B6E 00C9                <1>       add    cl, cl
00001B70 29CE                <1>         sub  esi, ecx  ; next line
00001B72 29CF                <1>         sub  edi, ecx
00001B74 59                  <1>         pop  ecx
                             <1>
00001B75 FECD                <1>       dec    ch      ; count of lines to move
00001B77 75E9                <1>       jnz    short n13 ; row loop
                             <1>       ; ch = 0
00001B79 88C6                <1>       mov    dh, al  ; #rows
                             <1> n14:
                             <1>       ; attribute in ah
00001B7B B020                <1>       mov    al, ' '       ; fill with blanks
                             <1> n15:
00001B7D E887FFFFFF          <1>       call   n11 ; 16/01/2016
                             <1>
00001B82 8A0D[C45E0000]      <1>       mov    cl, [CRT_COLS]
00001B88 00C9                <1>       add    cl, cl
00001B8A 29CF                <1>         sub  edi, ecx
                             <1>
00001B8C FECE                <1>       dec    dh
00001B8E 75ED                <1>       jnz    short n15
                             <1>       ;
00001B90 E9FDFEFFFF          <1>       jmp    n16 ; 27/06/2016
                             <1>
                             <1> ;     cmp    bh, [ACTIVE_PAGE]
                             <1> ;     jne    short n16
                             <1> ;
                             <1> ;     ;cmp   byte [CRT_MODE], 7  ; is this the black and white card
                             <1> ;     ;je    short n16           ; if so, skip the mode reset
                             <1> ;
                             <1> ;     mov    al, [CRT_MODE_SET] ; get the value of mode set
                             <1> ;     mov    dx, 03D8h ; always set color card port
                             <1> ;     out    dx, al
                             <1> ;n16:
                             <1> ;     ; !!!!
                             <1> ;     cld           ; Clear direction flag !
                             <1> ;     ; !!!!
                             <1> ;     retn
                             <1>
                             <1> READ_AC_CURRENT:
                             <1>       ; 08/07/2016
                             <1>       ; 26/06/2016
                             <1>       ; 12/05/2016
                             <1>       ; 18/01/2016
                             <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1>       ;
                             <1>       ; 08/07/2016
00001B95 803D[C25E0000]07    <1>         cmp     byte [CRT_MODE], 7 ; 6!?
00001B9C 7607                <1>       jna    short read_ac_c
00001B9E 31C0                <1>       xor    eax, eax
00001BA0 E9B4F9FFFF          <1>         jmp     _video_return
                             <1> read_ac_c:
00001BA5 E805000000          <1>       call   _read_ac_current
                             <1>       ; 12/05/2016
                             <1>         ;jmp     VIDEO_RETURN
00001BAA E9AAF9FFFF          <1>       jmp    _video_return
                             <1>
                             <1> ;------------------------------------------------------------------------
                             <1> ; READ_AC_CURRENT                                                    :
                             <1> ;     THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT     :
                             <1> ;     CURSOR POSITION AND RETURNS THEM TO THE CALLER                    :
                             <1> ; INPUT                                                                    :
                             <1> ;     (AH) = CURRENT CRT MODE                                          :
                             <1> ;     (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )                         :
                             <1> ;     (DS) = DATA SEGMENT                                       :
                             <1> ;     (ES) = REGEN SEGMENT                                             :
                             <1> ; OUTPUT                                                          :
                             <1> ;     (AL) = CHARACTER READ                                            :
                             <1> ;     (AH) = ATTRIBUTE READ                                            :
                             <1> ;------------------------------------------------------------------------
                             <1>
                             <1> _read_ac_current:
                             <1>       ; 26/06/2016
                             <1>       ; 12/05/2016
                             <1>       ; 18/01/2016
                             <1>
                             <1>       ;mov   ah, [CRT_MODE] ; current video mode
                             <1>       ;cmp   ah, 4
                             <1>       ;jb    short p10
```

```
00001BAF 803D[C25E0000]04    <1>       cmp    byte [CRT_MODE], 4
00001BB6 0F83BB080000        <1>       jnb     GRAPHICS_READ ; 26/06/2016
                             <1> 
                             <1>       ;cmp   ah, 7 ; TEST FOR BW CARD
                             <1>       ;jne   GRAPHICS_READ
                             <1> p10:
00001BBC E801FEFFFF          <1>       call   find_position; GET REGEN LOCATION AND PORT ADDRESS
                             <1>       ;
                             <1>       ; esi = regen location
                             <1>       ; dx = status port
                             <1>       ;
00001BC1 8A25[C25E0000]      <1>       mov    ah, [CRT_MODE]
00001BC7 80EC02              <1>       sub    ah, 2
00001BCA D0EC                <1>       shr    ah, 1
00001BCC 7515                <1>       jnz    short p13
                             <1> 
                             <1>       ; WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
                             <1> p11:
00001BCE FB                  <1>       sti          ; enable interrupts first
00001BCF 3A3D[66580100]      <1>       cmp     bh, [ACTIVE_PAGE]
00001BD5 750C                <1>       jne    short p13
00001BD7 FA                  <1>       cli          ; block interrupts for single loop
00001BD8 EC                  <1>       in     al, dx ; get status from the adapter
00001BD9 A801                <1>       test   al, RHRZ ; is horizontal retrace low
00001BDB 75F1                <1>       jnz    short p11 ; wait until it is
                             <1> p12:               ;  wait for either retrace high
00001BDD EC                  <1>       in     al, dx ; get status again
00001BDE A809                <1>       test   al, RVRT+RHRZ ; is horizontal or vertical retrace high
00001BE0 74FB                <1>       jz     short p12 ; wait until either retrace active
00001BE2 FB                  <1>       sti
                             <1> p13:
00001BE3 81C600800B00        <1>       add    esi, 0B8000h
00001BE9 668B06              <1>       mov    ax, [esi]
                             <1> 
00001BEC C3                  <1>       retn   ; 18/01/2016
                             <1> 
                             <1> WRITE_AC_CURRENT:
                             <1>       ; 08/07/2016
                             <1>       ; 26/06/2016
                             <1>       ; 24/06/2016
                             <1>       ; 12/05/2016
                             <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1>       ;
                             <1> ;----------------------------------------------------------------
                             <1> ; WRITE_AC_CURRENT                                  :
                             <1> ;     THTS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER            :
                             <1> ;     AT THE CURRENT CURSOR POSITION                       :
                             <1> ; INPUT                                                    :
                             <1> ;     (AH) = CURRENT CRT MODE                              :
                             <1> ;     (BH) = DISPLAY PAGE                           :
                             <1> ;     (CX) = COUNT OF CHARACTERS TO WRITE                  :
                             <1> ;     (AL) = CHAR TO WRITE                                 :
                             <1> ;     (BL) = ATTRIBUTE OF CHAR TO WRITE                    :
                             <1> ;     (DS) = DATA SEGMENT                           :
                             <1> ;     (ES) = REGEN SEGMENT                                 :
                             <1> ; OUTPUT                                            :
                             <1> ;     DISPLAY REGEN BUFFER UPDATED                         :
                             <1> ;----------------------------------------------------------------
                             <1> 
                             <1>       ; 08/07/2016
00001BED 803D[C25E0000]07    <1>       cmp    byte [CRT_MODE], 7 ; 6!?
00001BF4 760A                <1>       jna    short write_ac_c
                             <1> 
00001BF6 E8F20A0000          <1>       call   vga_write_char_attr
00001BFB E954F9FFFF          <1>       jmp     VIDEO_RETURN
                             <1> 
                             <1> write_ac_c:
00001C00 E834000000          <1>       call   _write_c_current
                             <1> 
00001C05 0FB6F7              <1>       movzx  esi, bh ; video page number (0 to 7)
00001C08 889E[CB5E0000]      <1>       mov    [esi+chr_attrib], bl ; color/attribute
                             <1> 
00001C0E E941F9FFFF          <1>         jmp     VIDEO_RETURN
                             <1> 
                             <1> WRITE_C_CURRENT:
                             <1>       ; 08/07/2016
                             <1>       ; 26/06/2016
                             <1>       ; 12/05/2016
                             <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1>       ;
                             <1> ;----------------------------------------------------------------
                             <1> ; WRITE_C_CURRENT                                   :
                             <1> ;     THIS ROUTINE WRITES THE CHARACTER AT                 :
                             <1> ;     THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED     :
                             <1> ; INPUT                                                    :
                             <1> ;     (AH) = CURRENT CRT MODE                              :
                             <1> ;     (BH) = DISPLAY PAGE                           :
                             <1> ;     (CX) = COUNT OF CHARACTERS TO WRITE                  :
                             <1> ;     (AL) = CHAR TO WRITE                                 :
                             <1> ;     (DS) = DATA SEGMENT                           :
                             <1> ;     (ES) = REGEN SEGMENT                                 :
                             <1> ; OUTPUT                                            :
                             <1> ;     DISPLAY REGEN BUFFER UPDATED                         :
                             <1> ;----------------------------------------------------------------
                             <1> 
                             <1>       ; 08/07/2016
00001C13 803D[C25E0000]07    <1>       cmp    byte [CRT_MODE], 7 ; 6!?
00001C1A 760A                <1>       jna    short write_c_c
                             <1> 
```

```
00001C1C E8CC0A0000          <1>       call   vga_write_char_only
00001C21 E92EF9FFFF          <1>       jmp     VIDEO_RETURN
                             <1>
                             <1> write_c_c:
                             <1>       ;and   bh, 7 ; video page number (<= 7)
00001C26 0FB6F7              <1>       movzx  esi, bh
00001C29 8A9E[CB5E0000]      <1>       mov    bl, [esi+chr_attrib]
                             <1>
00001C2F E805000000          <1>       call   _write_c_current
00001C34 E91BF9FFFF          <1>         jmp     VIDEO_RETURN
                             <1>
                             <1> _write_c_current:  ; from 'write_tty'
                             <1>       ; 26/06/2016
                             <1>       ; 24/06/2016
                             <1>       ; 12/05/2016
                             <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 30/08/2014 (Retro UNIX 386 v1)
                             <1>       ; 18/01/2014
                             <1>       ; 04/12/2013
                             <1>       ;
                             <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1>
                             <1>       ;mov   ah, [CRT_MODE] ; current video mode
                             <1>       ;cmp   ah, 4
                             <1>       ;jb    short p40
00001C39 803D[C25E0000]04    <1>       cmp    byte [CRT_MODE], 4
00001C40 0F8381070000        <1>         jnb     GRAPHICS_WRITE ; 26/06/2016
                             <1>
                             <1>       ;cmp   ah, 7 ; TEST FOR BW CARD
                             <1>       ;jne   GRAPHICS_WRITE
                             <1> p40:
                             <1>       ; al = character
                             <1>       ; bl = color/attribute
                             <1>       ; bh = video page
                             <1>       ; cx = count of characters to write
00001C46 6652                <1>       push   dx
00001C48 88DC                <1>       mov    ah, bl  ; color/attribute (12/05/2016)
00001C4A 6650                <1>       push   ax     ; save character & attribute/color
00001C4C 6651                <1>       push   cx
00001C4E E86FFDFFFF          <1>       call   find_position  ; get regen location and port address
00001C53 6659                <1>       pop    cx
                             <1>       ; esi = regen location
                             <1>       ; dx = status port
                             <1>       ;
00001C55 81C600800B00        <1>       add    esi, 0B8000h ; 30/08/2014 (crt_base)
                             <1>       ;
00001C5B 8A25[C25E0000]      <1>       mov    ah, [CRT_MODE]
00001C61 80EC02              <1>       sub    ah, 2
00001C64 D0EC                <1>       shr    ah, 1
00001C66 7519                <1>       jnz    short p44    ; 26/06/2016
                             <1>
                             <1>       ; WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
                             <1> p41:
00001C68 FB                  <1>       sti           ; enable interrupts first
00001C69 3A3D[66580100]      <1>       cmp      bh, [ACTIVE_PAGE]
00001C6F 7510                <1>       jne    short p44
00001C71 FA                  <1>       cli           ; block interrupts for single loop
00001C72 EC                  <1>       in     al, dx ; get status from the adapter
00001C73 A808                <1>       test   al, RVRT ; check for vertical retrace first
00001C75 7509                <1>       jnz    short p43 ; Do fast write now if vertical retrace
00001C77 A801                <1>       test   al, RHRZ ; is horizontal retrace low
00001C79 75ED                <1>       jnz    short p41 ; wait until it is
                             <1> p42:              ;  wait for either retrace high
00001C7B EC                  <1>       in     al, dx ; get status again
00001C7C A809                <1>       test   al, RVRT+RHRZ ; is horizontal or vertical retrace high
00001C7E 74FB                <1>       jz     short p42 ; wait until either retrace active
                             <1> p43:
00001C80 FB                  <1>       sti
                             <1> p44:
00001C81 668B0424            <1>       mov    ax, [esp] ; restore the character (al) & attribute (ah)
00001C85 668906              <1>       mov    [esi], ax
                             <1>
00001C88 6649                <1>       dec    cx
00001C8A 7404                <1>       jz     short p45
                             <1>
00001C8C 46                  <1>       inc    esi
00001C8D 46                  <1>       inc    esi
00001C8E EBD8                <1>       jmp    short p41
                             <1> p45:
00001C90 6658                <1>       pop    ax
00001C92 665A                <1>       pop    dx
00001C94 C3                  <1>       retn
                             <1>
                             <1> ; 09/07/2016
                             <1> ; 26/06/2016
                             <1> ; 24/06/2016
                             <1> ; 12/05/2016
                             <1> ; 18/01/2016
                             <1> ; 16/01/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> ; 30/06/2015
                             <1> ; 27/06/2015
                             <1> ; 11/03/2015
                             <1> ; 02/09/2014
                             <1> ; 30/08/2014
                             <1> ; VIDEO FUNCTIONS
                             <1> ; (write_tty - Retro UNIX 8086 v1 - U9.ASM, 01/02/2014)
                             <1>
                             <1> WRITE_TTY:
                             <1>       ; 09/12/2017
                             <1>       ; 09/07/2016
                             <1>       ; 01/07/2016
                             <1>       ; 26/06/2016
                             <1>       ; 24/06/2016
                             <1>        ; 13/05/2016
                             <1>        ; 12/05/2016
                             <1>        ; 30/01/2016
                             <1>        ; 18/01/2016
                             <1>        ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>        ; 13/08/2015
                             <1>        ; 02/09/2014
                             <1>        ; 30/08/2014 (Retro UNIX 386 v1 - beginning)
                             <1>        ; 01/02/2014 (Retro UNIX 8086 v1 - last update)
                             <1>        ; 03/12/2013 (Retro UNIX 8086 v1 - beginning)
                             <1>        ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
                             <1>        ;
                             <1>        ; INPUT -> AL = Character to be written
                             <1>        ;         BL = Color (Forecolor, Backcolor)
                             <1>        ;         BH = Video Page (0 to 7)
                             <1>
                             <1>        ; 09/07/2016
00001C95 803D[C25E0000]07    <1>          cmp     byte [CRT_MODE], 7
00001C9C 760A                <1>        jna    short write_tty_cga
                             <1>
00001C9E E8290D0000          <1>        call   vga_write_teletype
00001CA3 E9ACF8FFFF          <1>        jmp    VIDEO_RETURN
                             <1>
                             <1> write_tty_cga:
                             <1>        ; 13/05/2016
                             <1>        ;call _write_tty
                             <1>        ; 01/07/2016
00001CA8 E818000000          <1>        call   _write_tty_m3
00001CAD E9A2F8FFFF          <1>        jmp    VIDEO_RETURN
                             <1>
                             <1> RVRT  equ    00001000b    ; VIDEO VERTICAL RETRACE BIT
                             <1> RHRZ  equ    00000001b    ; VIDEO HORIZONTAL RETRACE BIT
                             <1>
                             <1> ; Derived from "WRITE_TTY" procedure of IBM "pc-at" rombios source code
                             <1> ; (06/10/1985), 'video.asm', INT 10H, VIDEO_IO
                             <1> ;
                             <1> ; 06/10/85  VIDEO DISPLAY BIOS
                             <1> ;
                             <1> ;--- WRITE_TTY ------------------------------------------------------------------
                             <1> ;                                                                :
                             <1> ;   THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE                  :
                             <1> ;   VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT                :
                             <1> ;   CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.            :
                             <1> ;   IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN             :
                             <1> ;   IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW       :
                             <1> ;   ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW,         :
                             <1> ;   FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.       :
                             <1> ;   WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE             :
                             <1> ;   NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS       :
                             <1> ;   LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE,       :
                             <1> ;   THE 0 COLOR IS USED.                                         :
                             <1> ;   ENTRY --                                                     :
                             <1> ;     (AH) = CURRENT CRT MODE                                          :
                             <1> ;     (AL) = CHARACTER TO BE WRITTEN                                   :
                             <1> ;         NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE :
                             <1> ;         HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS      :
                             <1> ;     (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE  :
                             <1> ;   EXIT --                                                      :
                             <1> ;     ALL REGISTERS SAVED                                        :
                             <1> ;--------------------------------------------------------------------------------
                             <1>
                             <1> ; 09/12/2017
                             <1> ; 08/07/2016
                             <1> ; 26/06/2016
                             <1> ; 24/06/2016
                             <1> _write_tty:  ; 13/05/2016
00001CB2 FA                  <1>        cli
                             <1>        ;
                             <1>        ; 01/09/2014
00001CB3 803D[C25E0000]03    <1>        cmp    byte [CRT_MODE], 3
00001CBA 7409                <1>        je     short _write_tty_m3
                             <1>        ;
                             <1> set_mode_3:
00001CBC 53                  <1>        push   ebx
00001CBD 50                  <1>        push   eax
00001CBE E8A2F8FFFF          <1>        call   _set_mode
00001CC3 58                  <1>        pop    eax
00001CC4 5B                  <1>        pop    ebx
                             <1>        ;
                             <1> _write_tty_m3: ; 24/06/2016 (m3 -> _write_tty_m3)
00001CC5 0FB6F7              <1>        movzx  esi, bh ; 12/05/2016
00001CC8 66D1E6              <1>        shl    si, 1
00001CCB 81C6[56580100]      <1>        add    esi, CURSOR_POSN
00001CD1 668B16              <1>        mov    dx, [esi]
                             <1>        ;
                             <1>        ; dx now has the current cursor position
                             <1>        ;
00001CD4 3C0D                <1>        cmp    al, 0Dh      ; CR   ; is it carriage return or control character
00001CD6 7636                <1>        jbe    short u8
                             <1>        ;
                             <1>        ; write the char to the screen
                             <1> u0:
                             <1>        ; al = character
                             <1>        ; bl = attribute/color
                             <1>        ; bh = video page number (0 to 7)
                             <1>        ;
00001CD8 66B90100            <1>        mov    cx, 1  ; 24/06/2016
                             <1>        ; cx = count of characters to write
                             <1>        ;
00001CDC E858FFFFFF          <1>        call   _write_c_current ; 16/01/2015
                             <1>        ;
                             <1>        ; position the cursor for next char
00001CE1 FEC2                <1>        inc    dl           ; next column
```

```
00001CE3 3A15[C45E0000]      <1>       cmp    dl, [CRT_COLS]  ; test for column overflow
00001CE9 755D                <1>         jne     _set_cpos
00001CEB B200                <1>       mov    dl, 0        ; column = 0
                             <1> u10:                       ; (line feed found)
00001CED 80FE18              <1>       cmp    dh, 25-1     ; check for last row
00001CF0 7218                <1>       jb     short u6
                             <1>       ;
                             <1>       ; scroll required
                             <1> u1:
                             <1>       ; SET CURSOR POSITION (04/12/2013)
00001CF2 E851000000          <1>       call   _set_cpos
                             <1>       ;
                             <1>       ; determine value to fill with during scroll
                             <1> u2:
                             <1>       ; bh = video page number
                             <1>       ;
00001CF7 E8B3FEFFFF          <1>       call   _read_ac_current ; 18/01/2016
                             <1>       ;
                             <1>       ; al = character, ah = attribute
                             <1>       ; bh = video page number
                             <1> u3:
                             <1>       ;;mov  ax, 0601h    ; scroll one line
                             <1>       ;;sub  cx, cx       ; upper left corner
                             <1>       ;;mov  dh, 25-1     ; lower right row
                             <1>       ;;;mov dl, [CRT_COLS]
                             <1>       ;mov   dl, 80       ; lower right column
                             <1>       ;;dec  dl
                             <1>       ;;mov  dl, 79
                             <1>
                             <1>       ;;call scroll_up    ; 04/12/2013
                             <1>       ;;; 11/03/2015
                             <1>       ; 02/09/2014
                             <1>       ;;;mov cx, [crt_ulc] ; Upper left corner  (0000h)
                             <1>       ;;;mov dx, [crt_lrc] ; Lower right corner (184Fh)
                             <1>       ; 11/03/2015
00001CFC 6629C9              <1>       sub    cx, cx
00001CFF 66BA4F18            <1>       mov    dx, 184Fh ; dl = 79 (column), dh = 24 (row)
                             <1>       ;
00001D03 B001                <1>       mov    al, 1        ; scroll 1 line up
                             <1>              ; ah = attribute
                             <1>       ;mov   bl, al ; 12/05/2016
00001D05 E900FDFFFF          <1>       jmp    _scroll_up   ; 16/01/2016
                             <1> ;u4:
                             <1>       ;;int  10h          ; video-call return
                             <1>                            ; scroll up the screen
                             <1>                            ; tty return
                             <1> ;u5:
                             <1>       ;retn               ; return to the caller
                             <1>
                             <1> u6:                        ; set-cursor-inc
00001D0A FEC6                <1>       inc    dh           ; next row
                             <1>                            ; set cursor
                             <1> ;u7:
                             <1>       ;;mov  ah, 02h
                             <1>       ;;jmp  short u4     ; establish the new cursor
                             <1>       ;call  _set_cpos
                             <1>       ;jmp   short u5
00001D0C EB3A                <1>       jmp      _set_cpos
                             <1>
                             <1>       ; check for control characters
                             <1> u8:
00001D0E 7436                <1>       je     short u9
00001D10 3C0A                <1>       cmp    al, 0Ah             ; is it a line feed (0Ah)
00001D12 74D9                <1>       je     short u10
00001D14 3C07                <1>       cmp    al, 07h      ; is it a bell
00001D16 747A                <1>       je     short u11
00001D18 3C08                <1>       cmp    al, 08h             ; is it a backspace
                             <1>       ;jne   short u0
00001D1A 7422                <1>       je     short bs     ; 12/12/2013
                             <1>       ; 12/12/2013 (tab stop)
00001D1C 3C09                <1>       cmp    al, 09h             ; is it a tab stop
00001D1E 75B8                <1>       jne    short u0
00001D20 88D0                <1>       mov    al, dl
                             <1>       ;cbw
00001D22 30E4                <1>       xor    ah, ah ; 09/12/2017
00001D24 B108                <1>       mov    cl, 8
00001D26 F6F1                <1>       div    cl
00001D28 28E1                <1>       sub    cl, ah
                             <1> ts:
                             <1>       ; 02/09/2014
                             <1>       ; 01/09/2014
00001D2A B020                <1>       mov    al, 20h
                             <1> tsloop:
00001D2C 6651                <1>       push   cx
00001D2E 6650                <1>       push   ax
                             <1>       ;mov   bh, [ACTIVE_PAGE]
00001D30 E890FFFFFF          <1>       call   _write_tty_m3 ; 24/06/2016 (m3 -> _write_tty_m3)
00001D35 6658                <1>       pop    ax  ; ah = attribute/color
00001D37 6659                <1>       pop    cx
00001D39 FEC9                <1>       dec    cl
00001D3B 75EF                <1>       jnz    short tsloop
00001D3D C3                  <1>       retn
                             <1> bs:
                             <1>       ; back space found
                             <1>
00001D3E 08D2                <1>       or     dl, dl              ; is it already at start of line
                             <1>       ;je    short u7     ; set_cursor
00001D40 7406                <1>       jz     short _set_cpos
00001D42 664A                <1>       dec    dx                  ; no -- just move it back
                             <1>       ;jmp   short u7
00001D44 EB02                <1>       jmp    short _set_cpos
                             <1>
                             <1>       ; carriage return found
                             <1> u9:
00001D46 B200                <1>       mov    dl, 0        ; move to first column
                             <1>       ;jmp   short u7
                             <1>       ;jmp   short _set_cpos ; 30/01/2016
                             <1>
                             <1>       ; line feed found
                             <1> ;u10:
                             <1> ;     cmp    dh, 25-1     ; bottom of screen
                             <1> ;     jne    short u6     ; no, just set the cursor
                             <1> ;       jmp     u1              ; yes, scroll the screen
                             <1>
                             <1> _set_cpos:
                             <1>       ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 27/06/2015
                             <1>       ; 01/09/2014
                             <1>       ; 30/08/2014 (Retro UNIX 386 v1)
                             <1>       ;
                             <1>       ; 04/12/2013 - 12/12/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1>       ;
                             <1> ;----------------------------------------------
                             <1> ; SET_CPOS
                             <1> ;     THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
                             <1> ;     NEW X-Y VALUES PASSED
                             <1> ; INPUT
                             <1> ;     DX - ROW,COLUMN OF NEW CURSOR
                             <1> ;     BH - DISPLAY PAGE OF CURSOR
                             <1> ; OUTPUT
                             <1> ;     CURSOR ID SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
                             <1> ;----------------------------------------------
                             <1>       ;
00001D48 BE[56580100]        <1>       mov    esi, CURSOR_POSN
00001D4D 0FB6C7              <1>         movzx   eax, bh   ; BH = video page number
                             <1> ;     or     al, al
                             <1> ;     jz     short _set_cpos_0
00001D50 D0E0                <1>         shl     al, 1   ; word offset
00001D52 01C6                <1>         add     esi, eax
                             <1> ;_set_cpos_0:
00001D54 668916              <1>       mov    [esi], dx ; save the pointer
00001D57 383D[66580100]      <1>       cmp    [ACTIVE_PAGE], bh
00001D5D 7532                <1>       jne    short m17
                             <1>       ;call  m18    ; CURSOR SET
                             <1> ;m17:               ; SET_CPOS_RETURN
                             <1>       ; 01/09/2014
                             <1> ;     retn
                             <1>              ; DX  = row/column
                             <1> m18:
00001D5F E84CFCFFFF          <1>       call   position ; determine location in regen buffer
00001D64 668B0D[54580100]    <1>       mov    cx, [CRT_START]
00001D6B 6601C1              <1>       add    cx, ax  ; add char position in regen buffer
                             <1>                     ; to the start address (offset) for this page
00001D6E 66D1E9              <1>       shr    cx, 1  ; divide by 2 for char only count
00001D71 B40E                <1>       mov    ah, 14 ; register number for cursor
                             <1>       ;call  m16    ; output value to the 6845
                             <1>       ;retn
                             <1>
                             <1>       ;----- THIS ROUTINE OUTPUTS THE CX REGISTER
                             <1>       ;      TO THE 6845 REGISTERS NAMED IN (AH)
                             <1> m16:
00001D73 FA                  <1>       cli
                             <1>       ;mov   dx, [addr_6845] ; address register
00001D74 66BAD403            <1>       mov    dx, 03D4h ; I/O address of color card
00001D78 88E0                <1>       mov    al, ah ; get value
00001D7A EE                  <1>       out    dx, al ; register set
00001D7B 6642                <1>       inc    dx     ; data register
00001D7D EB00                <1>       jmp    $+2    ; i/o delay
00001D7F 88E8                <1>       mov    al, ch ; data
00001D81 EE                  <1>       out    dx, al
00001D82 664A                <1>       dec    dx
00001D84 88E0                <1>       mov    al, ah
00001D86 FEC0                <1>       inc    al     ; point to other data register
00001D88 EE                  <1>       out    dx, al ; set for second register
00001D89 6642                <1>       inc    dx
00001D8B EB00                <1>       jmp    $+2    ; i/o delay
00001D8D 88C8                <1>       mov    al, cl ; second data value
00001D8F EE                  <1>       out    dx, al
00001D90 FB                  <1>       sti
                             <1> m17:
00001D91 C3                  <1>       retn
                             <1>
                             <1> beeper:
                             <1>       ; 04/08/2016
                             <1>       ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 30/08/2014 (Retro UNIX 386 v1)
                             <1>       ; 18/01/2014
                             <1>       ; 03/12/2013
                             <1>       ; bell found
                             <1> u11:
00001D92 FB                  <1>       sti
00001D93 3A3D[66580100]      <1>       cmp    bh, [ACTIVE_PAGE]
00001D99 7551                <1>       jne    short u12    ; Do not sound the beep
                             <1>                           ; if it is not written on the active page
                             <1> beeper_gfx: ; 04/08/2016
00001D9B 66B93305            <1>       mov    cx, 1331     ; divisor for 896 hz tone
00001D9F B31F                <1>       mov    bl, 31       ; set count for 31/64 second for beep
                             <1>       ;call  beep         ; sound the pod bell
                             <1>       ;jmp   short u5     ; tty_return
                             <1>       ;retn
                             <1>
                             <1> TIMER equ    040h                ; 8254 TIMER - BASE ADDRESS
                             <1> PORT_B       equ    061h          ; PORT B READ/WRITE DIAGNOSTIC REGISTER
                             <1> GATE2 equ    00000001b    ; TIMER 2 INPUT CATE CLOCK BIT
                             <1> SPK2  equ    00000010b    ; SPEAKER OUTPUT DATA ENABLE BIT
                             <1> 
                             <1> beep:
                             <1>       ; 07/02/2015
                             <1>       ; 30/08/2014 (Retro UNIX 386 v1)
                             <1>       ; 18/01/2014
                             <1>       ; 03/12/2013
                             <1>       ;
                             <1>       ; TEST4.ASM - 06/10/85  POST AND BIOS UTILITY ROUTINES
                             <1>       ;
                             <1>       ; ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE
                             <1>       ;
                             <1>       ; ENTRY:
                             <1>       ;    (BL) = DURATION COUNTER ( 1 FOR 1/64 SECOND )
                             <1>       ;    (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ)
                             <1>       ; EXIT:                          :
                             <1>       ;    (AX),(BL),(CX) MODIFIED.
                             <1> 
00001DA1 9C                  <1>       pushf  ; 18/01/2014 ; save interrupt status
00001DA2 FA                  <1>       cli                 ; block interrupts during update
00001DA3 B0B6                <1>       mov    al, 10110110b; select timer 2, lsb, msb binary
00001DA5 E643                <1>       out    TIMER+3, al  ; write timer mode register
00001DA7 EB00                <1>       jmp    $+2          ; I/O delay
00001DA9 88C8                <1>       mov    al, cl       ; divisor for hz (low)
00001DAB E642                <1>       out    TIMER+2,AL   ; write timer 2 count - lsb
00001DAD EB00                <1>       jmp    $+2          ; I/O delay
00001DAF 88E8                <1>       mov    al, ch       ; divisor for hz (high)
00001DB1 E642                <1>       out    TIMER+2, al  ; write timer 2 count - msb
00001DB3 E461                <1>       in     al, PORT_B   ; get current setting of port
00001DB5 88C4                <1>       mov    ah, al       ; save that setting
00001DB7 0C03                <1>       or     al, GATE2+SPK2      ; gate timer 2 and turn speaker on
00001DB9 E661                <1>       out    PORT_B, al   ; and restore interrupt status
                             <1>       ;popf  ; 18/01/2014
00001DBB FB                  <1>       sti
                             <1> g7:                       ; 1/64 second per count (bl)
00001DBC B90B040000          <1>       mov    ecx, 1035    ; delay count for 1/64 of a second
00001DC1 E827000000          <1>       call   waitf        ; go to beep delay 1/64 count
00001DC6 FECB                <1>       dec    bl           ; (bl) length count expired?
00001DC8 75F2                <1>       jnz    short g7     ; no - continue beeping speaker
                             <1>       ;
                             <1>       ;pushf              ; save interrupt status
00001DCA FA                  <1>       cli    ; 18/01/2014 ; block interrupts during update
00001DCB E461                <1>       in     al, PORT_B   ; get current port value
                             <1>         ;or      al, not (GATE2+SPK2) ; isolate current speaker bits in case
00001DCD 0CFC                <1>         or      al, ~(GATE2+SPK2)
00001DCF 20C4                <1>         and ah, al        ; someone turned them off during beep
00001DD1 88E0                <1>       mov    al, ah       ; recover value of port
                             <1>         ;or      al, not (GATE2+SPK2) ; force speaker data off
00001DD3 0CFC                <1>       or     al, ~(GATE2+SPK2) ; isolate current speaker bits in case
00001DD5 E661                <1>       out    PORT_B, al   ; and stop speaker timer
                             <1>       ;popf               ; restore interrupt flag state
00001DD7 FB                  <1>       sti
00001DD8 B90B040000          <1>       mov    ecx, 1035    ; force 1/64 second delay (short)
00001DDD E80B000000          <1>       call   waitf        ; minimum delay between all beeps
                             <1>       ;pushf              ; save interrupt status
00001DE2 FA                  <1>       cli                 ; block interrupts during update
00001DE3 E461                <1>       in     al, PORT_B   ; get current port value in case
00001DE5 2403                <1>       and    al, GATE2+SPK2      ; someone turned them on
00001DE7 08E0                <1>       or     al, ah       ; recover value of port_b
00001DE9 E661                <1>       out    PORT_B, al   ; restore speaker status
00001DEB 9D                  <1>       popf                ; restore interrupt flag state
                             <1> u12:
00001DEC C3                  <1>       retn
                             <1> 
                             <1> REFRESH_BIT equ    00010000b    ; REFRESH TEST BIT
                             <1> 
                             <1> WAITF:
                             <1> waitf:
                             <1>       ; 30/08/2014 (Retro UNIX 386 v1)
                             <1>       ; 03/12/2013
                             <1>       ;
                             <1> ;     push ax                   ; save work register (ah)
                             <1> ;waitf1:
                             <1>                           ; use timer 1 output bits
                             <1> ;     in     al, PORT_B   ; read current counter output status
                             <1> ;     and    al, REFRESH_BIT     ; mask for refresh determine bit
                             <1> ;     cmp    al, ah       ; did it just change
                             <1> ;     je     short waitf1 ; wait for a change in output line
                             <1> ;     ;
                             <1> ;     mov    ah, al       ; save new lflag state
                             <1> ;     loop   waitf1       ; decrement half cycles till count end
                             <1> ;     ;
                             <1> ;     pop    ax           ; restore (ah)
                             <1> ;     retn                ; return (cx)=0
                             <1> 
                             <1> ; 06/02/2015 (unix386.s <-- dsectrm2.s)
                             <1> ; 17/12/2014 (dsectrm2.s)
                             <1> ; WAITF
                             <1> ; /// IBM PC-XT Model 286 System BIOS Source Code - Test 4 - 06/10/85 ///
                             <1> ;
                             <1> ;---WAITF-----------------------------------------------------------------------
                             <1> ;     FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR)
                             <1> ; ENTRY:
                             <1> ;     (CX) = COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT
                             <1> ;           MEMORY REFRESH TIMER 1 OUTPUT USED AS REFERENCE
                             <1> ; EXIT:
                             <1> ;                 AFTER (CX) TIME COUNT (PLUS OR MINUS 16 MICROSECONDS)
                             <1> ;     (CX) = 0
                             <1> ;-------------------------------------------------------------------------------
                             <1> 
                             <1> ; Refresh period: 30 micro seconds (15-80 us)
                             <1> ; (16/12/2014 - AWARDBIOS 1999 - ATORGS.ASM, WAIT_REFRESH)
                             <1> 
                             <1> ;WAITF:                                ; DELAY FOR (CX)*15.085737 US
00001DED 6650                <1>       PUSH  AX                  ; SAVE WORK REGISTER (AH)
                             <1>       ; 16/12/2014
                             <1>       ;shr  cx, 1               ; convert to count of 30 micro seconds
00001DEF D1E9                <1>       shr   ecx, 1 ; 21/02/2015
                             <1> ;17/12/2014
                             <1> ;WAITF1:
                             <1> ;     IN    AL, PORT_B   ;061h  ; READ CURRENT COUNTER OUTPUT STATUS
                             <1> ;     AND   AL, REFRESH_BIT     ;00010000b ; MASK FOR REFRESH DETERMINE BIT
                             <1> ;     CMP   AL, AH              ; DID IT JUST CHANGE
                             <1> ;     JE    short WAITF1        ; WAIT FOR A CHANGE IN OUTPUT LINE
                             <1> ;     MOV   AH, AL              ; SAVE NEW FLAG STATE
                             <1> ;     LOOP  WAITF1              ; DECREMENT HALF CYCLES TILL COUNT END
                             <1>       ;
                             <1>       ; 17/12/2014
                             <1>       ;
                             <1>       ; Modification from 'WAIT_REFRESH' procedure of AWARD BIOS - 1999
                             <1>       ;
                             <1> ;WAIT_REFRESH:  Uses port 61, bit 4 to have CPU speed independent waiting.
                             <1> ;     INPUT:  CX = number of refresh periods to wait
                             <1> ;                   (refresh periods = 1 per 30 microseconds on most machines)
                             <1> WR_STATE_0:
00001DF1 E461                <1>       IN    AL,PORT_B           ; IN AL,SYS1
00001DF3 A810                <1>       TEST  AL,010H
00001DF5 74FA                <1>       JZ    SHORT WR_STATE_0
                             <1> WR_STATE_1:
00001DF7 E461                <1>       IN    AL,PORT_B           ; IN AL,SYS1
00001DF9 A810                <1>       TEST  AL,010H
00001DFB 75FA                <1>       JNZ   SHORT WR_STATE_1
00001DFD E2F2                <1>         LOOP    WR_STATE_0
                             <1>       ;
00001DFF 6658                <1>       POP   AX                  ; RESTORE (AH)
00001E01 C3                  <1>       RETn                      ; (CX) = 0
                             <1> 
                             <1> ; 09/07/2016
                             <1> ; 01/07/2016
                             <1> ; 24/06/2016
                             <1> ; 23/06/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; WRITE_STRING                                                          :
                             <1> ;     THIS ROUTINE WRITES A STRING OF CHARACTERS TO THE CRT.                   :
                             <1> ; INPUT                                                           :
                             <1> ;     (AL) = WRITE STRING COMMAND  0 - 3                                :
                             <1> ;     (BH) = DISPLAY PAGE (ACTIVE PAGE)                                 :
                             <1> ;     (CX) = COUNT OF CHARACTERS TO WRITE, IF (CX) = 0 THEN RETURN             :
                             <1> ;     (DX) = CURSOR POSITION FOR START OF STRING WRITE                  :
                             <1> ;     (BL) = ATTRIBUTE OF CHARACTER TO WRITE IF (AL) = 0  OR     (AL) = 1       :
                             <1> ;     (eBP) = SOURCE STRING OFFSET                                      :
                             <1> ; OUTPUT                                                          :
                             <1> ;     NONE                                                        :
                             <1> ;-------------------------------------------------------------------------------
                             <1> 
                             <1> ; AL = 00h: Assign all characters the attribute in BL; do not update cursor
                             <1> ; AL = 01h: Assign all characters the attribute in BL; update cursor
                             <1> ; AL = 02h: Use attributes in string; do not update cursor
                             <1> ; AL = 03h: Use attributes in string; update cursor
                             <1> 
                             <1> WRITE_STRING:
                             <1>       ; 12/09/2016
                             <1>       ; 09/07/2016
                             <1>       ;cmp  byte [CRT_MODE], 7 ; 6?!
                             <1>       ;ja   VIDEO_RETURN        ; not a valid function for VGA modes
                             <1>       ;
00001E02 A2[D0650100]        <1>       mov   [w_str_cmd], al           ; save (AL) command
00001E07 3C04                <1>       CMP   AL, 4               ; TEST FOR INVALID WRITE STRING OPTION
00001E09 0F8345F7FFFF        <1>       JNB   VIDEO_RETURN        ; IF OPTION INVALID THEN RETURN
                             <1> 
                             <1>         ;JCXZ   VIDEO_RETURN            ; IF ZERO LENGTH STRING THEN RETURN
                             <1> 
00001E0F 67E35E              <1>         jcxz    P55                    ; 01/07/2016
                             <1> 
                             <1> 
                             <1>       ; 01/07/2016
                             <1>       ;and  ecx, 0FFFFh
                             <1>       ; ECX = byte count
                             <1>       ;push ecx
00001E12 89EE                <1>       mov   esi, ebp ; user buffer
00001E14 BF00000700          <1>       mov   edi, Cluster_Buffer  ; system buffer
00001E19 E8A5C90000          <1>       call  transfer_from_user_buffer
                             <1>       ;pop  ecx
00001E1E 0F8230F7FFFF        <1>       jc    VIDEO_RETURN
                             <1>       ; ecx = transfer (byte) count = character count
00001E24 BD00000700          <1>       mov   ebp, Cluster_Buffer
                             <1>       ; 12/09/2016
00001E29 803D[C25E0000]07    <1>       cmp   byte [CRT_MODE], 7 ; 6?!
00001E30 0F879F000000        <1>       ja    vga_write_string
                             <1>       ;
00001E36 0FB6F7              <1>       movzx esi, bh                    ; GET CURRENT CURSOR PAGE
00001E39 66D1E6              <1>       SAL   SI,1                ; CONVERT TO PAGE OFFSET  (SI= PAGE)
                             <1>       ; *****
00001E3C 66FFB6[56580100]    <1>       PUSH  word [eSI+CURSOR_POSN]     ; SAVE CURRENT CURSOR POSITION IN STACK
                             <1> 
                             <1>       ;MOV  AX,0200H            ; SET NEW CURSOR POSITION
                             <1>       ;INT  10H
                             <1> P50next:
00001E43 53                  <1>       push  ebx ; ****
00001E44 51                  <1>       push  ecx ; ***
00001E45 56                  <1>       push  esi ; **
00001E46 52                  <1>       push  edx ; *
00001E47 E8FCFEFFFF          <1>       call  _set_cpos
                             <1> P50:
00001E4C 8A4500              <1>       MOV   AL, [eBP]           ; GET CHARACTER FROM INPUT STRING
```

```
00001E4F 45                  <1>       INC    eBP                 ; BUMP POINTER TO CHARACTER
                             <1>
                             <1> ;-----       TEST FOR SPECIAL CHARACTER'S
                             <1>
00001E50 3C08                <1>       CMP    AL, 08H                   ; IS IT A BACKSPACE
00001E52 740C                <1>       JE     short P51           ; BACK_SPACE
00001E54 3C0D                <1>       CMP    AL, 0Dh ; CR        ; IS IT CARRIAGE RETURN
00001E56 7408                <1>       JE     short P51           ; CAR_RET
00001E58 3C0A                <1>       CMP    AL, 0Ah ; LF        ; IS IT A LINE FEED
00001E5A 7404                <1>       JE     short P51           ; LINE_FEED
00001E5C 3C07                <1>       CMP    AL, 07h                   ; IS IT A BELL
00001E5E 7515                <1>       JNE    short P52           ; IF NOT THEN DO WRITE CHARACTER
                             <1> P51:
                             <1>       ;MOV   AH,0EH              ; TTY_CHARACTER_WRITE
                             <1>       ;INT   10H                 ; WRITE TTY CHARACTER TO THE CRT
                             <1>
00001E60 E860FEFFFF          <1>       call   _write_tty_m3
                             <1>
00001E65 5A                  <1>       pop    edx ; *
00001E66 5E                  <1>       pop    esi ; **
                             <1>
00001E67 668B96[56580100]    <1>       MOV    DX, [eSI+CURSOR_POSN]     ; GET CURRENT CURSOR POSITION
00001E6E EB46                <1>       JMP    SHORT P54           ; SET CURSOR POSITION AND CONTINUE
                             <1> P55:
00001E70 E9DFF6FFFF          <1>       JMP    VIDEO_RETURN
                             <1> P52:
00001E75 66B90100            <1>       MOV    CX, 1               ; SET CHARACTER WRITE AMOUNT TO ONE
00001E79 803D[D0650100]02    <1>       CMP    byte [w_str_cmd], 2 ; IS THE ATTRIBUTE IN THE STRING
00001E80 7204                <1>       JB     short P53           ; IF NOT THEN SKIP
00001E82 8A5D00              <1>       MOV    BL, [eBP]           ; ELSE GET NEW ATTRIBUTE
00001E85 45                  <1>       INC    eBP                 ; BUMP STRING POINTER
                             <1> P53:
                             <1>       ;MOV   AH,09H              ; GOT_CHARACTER
                             <1>       ;INT   10H                 ; WRITE CHARACTER TO THE CRT
                             <1>
00001E86 E8AEFDFFFF          <1>       call   _write_c_current
                             <1>
00001E8B 5A                  <1>       pop    edx ; *
                             <1>
00001E8C 0FB6F7              <1>       movzx  esi, bh ; video page number (0 to 7)
00001E8F 889E[CB5E0000]      <1>       mov    [esi+chr_attrib], bl ; color/attribute
                             <1>
00001E95 FEC2                <1>       INC    DL                  ; INCREMENT COLUMN COUNTER
00001E97 3A15[C45E0000]      <1>       CMP    DL, [CRT_COLS]            ; IF COLS ARE WITHIN RANGE FOR THIS MODE
00001E9D 7217                <1>       JB     short P54           ;    THEN GO TO COLUMNS SET
00001E9F FEC6                <1>       INC    DH                  ; BUMP ROW COUNTER BY ONE
00001EA1 28D2                <1>       SUB    DL, DL              ; SET COLUMN COUNTER TO ZERO
00001EA3 80FE19              <1>       CMP    DH, 25              ; IF ROWS ARE LESS THAN 25 THEN
00001EA6 720E                <1>       JB     short P54           ; GO TO ROWS_COLUMNS_SET
                             <1>
00001EA8 66B80A0E            <1>       MOV    AX,0E0AH            ; ELSE SCROLL SCREEN
                             <1>       ;INT   10H                 ; RESET ROW COUNTER TO 24
                             <1>
00001EAC E814FEFFFF          <1>       call   _write_tty_m3
                             <1>
00001EB1 66BA0018            <1>       mov    dx, 1800h           ; Column = 0, Row = 24
00001EB5 5E                  <1>       pop    esi ; **
                             <1> P54:
                             <1>                                   ; ROW_COLUMNS_SET
                             <1>       ;MOV   AX,0200H            ; SET NEW CURSOR POSITION COMMAND
                             <1>       ;INT   10H                 ; ESTABLISH NEW CURSOR POSITION
                             <1>
00001EB6 59                  <1>       pop    ecx ; ***
00001EB7 5B                  <1>       pop    ebx ; ****
                             <1>
                             <1>       ;LOOP  P50                 ; DO IT ONCE MORE UNTIL (CX) = ZERO
00001EB8 6649                <1>       dec    cx
00001EBA 7587                <1>       jnz    short P50next
                             <1>
00001EBC 665A                <1>       POP    DX  ; *****         ; RESTORE OLD CURSOR COORDINATES
                             <1>
00001EBE F605[D0650100]01    <1>       test   byte [w_str_cmd], 1 ; IF CURSOR WAS NOT TO BE MOVED
00001EC5 0F8589F6FFFF        <1>       JNZ    VIDEO_RETURN        ; THEN EXIT WITHOUT RESETTING OLD VALUE
                             <1>
                             <1>       ;MOV   AX,0200H            ; ELSE RESTORE OLD CURSOR POSITION
                             <1>       ;INT   10H
                             <1>                                   ; DONE - EXIT WRITE STRING
00001ECB E878FEFFFF          <1>       call   _set_cpos
00001ED0 E97FF6FFFF          <1>       JMP    VIDEO_RETURN        ; RETURN TO CALLER
                             <1>
                             <1> vga_write_string:
                             <1>       ; 12/09/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', ' biosfn_write_string'
                             <1>
                             <1>       ; INPUT                                                     :
                             <1>       ;      (AL) = WRITE STRING COMMAND  0 - 3                         :
                             <1>       ;      (BH) = DISPLAY PAGE (ACTIVE PAGE)                          :
                             <1>       ;      (CX) = COUNT OF CHARACTERS TO WRITE, IF (CX) = 0 THEN RETURN       :
                             <1>       ;      (DX) = CURSOR POSITION FOR START OF STRING WRITE           :
                             <1>       ;      (BL) = ATTRIBUTE OF CHARACTER TO WRITE IF (AL) = 0  OR     (AL) = 1  :
                             <1>       ;      (eBP) = SOURCE STRING OFFSET                               :
                             <1>       ; OUTPUT                                                    :
                             <1>       ;      NONE                                                 :
                             <1>       ;---------------------------------------------------------------------------;
                             <1>
                             <1>       ; AL = 00h: Assign all characters the attribute in BL; do not update cursor
                             <1>       ; AL = 01h: Assign all characters the attribute in BL; update cursor
                             <1>       ; AL = 02h: Use attributes in string; do not update cursor
                             <1>       ; AL = 03h: Use attributes in string; update cursor
```

```
                           <1>
                           <1>        ; biosfn_write_string(GET_AL(),GET_BH(),GET_BL(),CX,GET_DH(),GET_DL(),ES,BP);
                           <1>        ; static void biosfn_write_string (flag,page,attr,count,row,col,seg,offset)
                           <1>
                           <1>        ; // Read curs info for the page
                           <1>        ; biosfn_get_cursor_pos(page,&dummy,&oldcurs);
                           <1>        ; bh = video page = 0
                           <1>        ;movzx esi, word [CURSOR_POSN] ; current cursor position for video page 0
                           <1>
                           <1>        ; // if row=0xff special case : use current cursor position
                           <1>        ; if(row==0xff)
                           <1>        ;  {col=oldcurs&0x00ff;
                           <1>        ;   row=(oldcurs&0xff00)>>8;
                           <1>        ;  }
                           <1>
                           <1>        ;mov   al, [w_str_cmd]
                           <1>
00001ED5 80FEFF            <1>        cmp    dh, 0FFh
00001ED8 7407              <1>        je     short vga_wstr_1 ; user current cursor position
                           <1> vga_wstr_0:
                           <1>        ; set cursor position
00001EDA 668915[56580100]  <1>        mov    [CURSOR_POSN], dx ; save cursor pos for pg 0
                           <1> vga_wstr_1:
00001EE1 66FF35[56580100]  <1>        push   word [CURSOR_POSN] ; *
                           <1>
                           <1>        ; ebp = string offset in system buffer (user buffer was copied to)
                           <1>
                           <1>        ; while(count--!=0)
                           <1>        ;  {
                           <1>        ;   car=read_byte(seg,offset++);
                           <1>        ;   if((flag&0x02)!=0)
                           <1>        ;    attr=read_byte(seg,offset++);
                           <1>        ;    biosfn_write_teletype(car,page,attr,WITH_ATTR);
                           <1>        ;  }
                           <1>
                           <1>        ;push  eax ; **
                           <1>        ;test  al, 2
00001EE8 F605[D0650100]02  <1>        test   byte [w_str_cmd], 2
00001EEF 751D              <1>        jnz    short vga_wstr_3
00001EF1 881D[67580100]    <1>        mov    [ccolor], bl
                           <1> vga_wstr_2:
00001EF7 51                <1>        push   ecx
00001EF8 8A4500            <1>        mov    al, [ebp]
00001EFB E8CC0A0000        <1>        call   vga_write_teletype
00001F00 59                <1>        pop    ecx
00001F01 6649              <1>        dec    cx
00001F03 741E              <1>        jz     short vga_wstr_4
00001F05 45                <1>        inc    ebp
00001F06 8A1D[67580100]    <1>        mov    bl, [ccolor]
00001F0C EBE9              <1>        jmp    short vga_wstr_2
                           <1> vga_wstr_3:
00001F0E 51                <1>        push   ecx
00001F0F 8A4500            <1>        mov    al, [ebp]
00001F12 45                <1>        inc    ebp
00001F13 8A5D00            <1>        mov    bl, [ebp]
00001F16 E8B10A0000        <1>        call   vga_write_teletype
00001F1B 59                <1>        pop    ecx
00001F1C 6649              <1>        dec    cx
00001F1E 7403              <1>        jz     short vga_wstr_4
00001F20 45                <1>        inc    ebp
00001F21 EBEB              <1>        jmp    short vga_wstr_3
                           <1> vga_wstr_4:
                           <1>        ; // Set back curs pos
                           <1>        ; if((flag&0x01)==0)
                           <1>        ;  biosfn_set_cursor_pos(page,oldcurs);
                           <1>        ; }
                           <1>        ;pop   eax ; **
00001F23 665A              <1>        pop    dx ; word [CURSOR_POSN] ; *
                           <1>        ;test  al, 1
00001F25 F605[D0650100]01  <1>        test   byte [w_str_cmd], 1
00001F2C 0F8522F6FFFF      <1>        jnz    VIDEO_RETURN
00001F32 668915[56580100]  <1>        mov    [CURSOR_POSN], dx
00001F39 E916F6FFFF        <1>        JMP    VIDEO_RETURN
                           <1>
                           <1> ; 07/07/2016
                           <1> ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)
                           <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                           <1> ;-----------------------------------------------------
                           <1> ;  SCROLL UP
                           <1> ;   THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
                           <1> ; ENTRY ---
                           <1> ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                           <1> ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                           <1> ;   BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                           <1> ;  BH = FILL VALUE FOR BLANKED LINES
                           <1> ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                           <1> ;  DS = DATA SEGMENT
                           <1> ;  ES = REGEN SEGMENT
                           <1> ; EXIT --
                           <1> ;  NOTHING, THE SCREEN IS SCROLLED
                           <1> ;-------------------------------------------------------
                           <1>
                           <1>        ; cl = upper left column
                           <1>        ; ch = upper left row
                           <1>        ; dl = lower rigth column
                           <1>        ; dh = lower right row
                           <1>        ;
                           <1>        ; al = line count (AL=0 means blank entire fields)
                           <1>        ; bl = fill value for blanked lines
                           <1>        ; bh = unused
                           <1>
                           <1> GRAPHICS_UP:
                           <1>        ; 07/07/2016
```

```
                             <1>       ;AH = Current video mode, [CRT_MODE]
00001F3E 80FC07              <1>       cmp   ah, 7
00001F41 7766                <1>       ja    short vga_graphics_up
                             <1>       ;je   n0
                             <1>
00001F43 88C7                <1>       MOV   bh, al              ; save line count in BH
00001F45 6689C8              <1>       MOV   AX, CX              ; GET UPPER LEFT POSITION INTO AX REG
                             <1>
                             <1> ;-----      USE CHARACTER SUBROUTINE FOR POSITIONING
                             <1> ;-----      ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
                             <1>
00001F48 E8D9050000          <1>       CALL  GRAPH_POSN
00001F4D 0FB7F8              <1>       MOVzx eDI, AX                    ; SAVE RESULT AS DESTINATION ADDRESS
                             <1>
                             <1> ;-----      DETERMINE SIZE OF WINDOW
                             <1>
00001F50 6629CA              <1>       SUB   DX, CX
00001F53 6681C20101          <1>         ADD     DX, 101h                ; ADJUST VALUES
00001F58 C0E602              <1>       SAL   DH, 2               ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
                             <1>                                 ; AND EVEN/ODD ROWS
                             <1> ;-----      DETERMINE CRT MODE
                             <1>
00001F5B 803D[C25E0000]06    <1>       CMP   byte [CRT_MODE], 6  ; TEST FOR MEDIUM RES
00001F62 7305                <1>         JNC     short _R7_              ; FIND_SOURCE
                             <1>
                             <1> ;-----      MEDIUM RES UP
00001F64 D0E2                <1>       SAL   DL, 1               ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
00001F66 66D1E7              <1>       SAL   DI, 1               ; OFFSET *2 SINCE 2 BYTES/CHAR
                             <1>
                             <1> ;-----      DETERMINE THE SOURCE ADDRESS IN THE BUFFER
                             <1> _R7_:                                   ; FIND_SOURCE
00001F69 81C700800B00        <1>       add   edi, 0B8000h
00001F6F C0E702              <1>       sal   bh, 2               ; multiply number of lines by 4
00001F72 7431                <1>         JZ      short _R11              ; IF ZERO, THEN BLANK ENTIRE FIELD
00001F74 B050                <1>       MOV   AL, 80              ; 80 BYTES/ROW
00001F76 F6E7                <1>       mul   bh                  ; determine offset to source
00001F78 0FB7F0              <1>       movzx esi, ax                    ; offset to source
00001F7B 01FE                <1>       add   eSI, eDI            ; SET UP SOURCE
00001F7D 88F4                <1>       MOV   AH, DH              ; NUMBER OF ROWS IN FIELD
00001F7F 28FC                <1>       sub   ah, bh              ; determine number to move
                             <1>
                             <1> ;-----      LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
                             <1> _R8:                                    ; ROW_LOOP
00001F81 E812040000          <1>         CALL    _R17                    ; MOVE ONE ROW
00001F86 6681EEB01F          <1>       SUB   SI, 2000h-80        ; MOVE TO NEXT ROW
00001F8B 6681EFB01F          <1>       SUB   DI, 2000h-80
00001F90 FECC                <1>       DEC   AH                  ; NUMBER OF ROWS TO MOVE
00001F92 75ED                <1>         JNZ     short _R8               ; CONTINUE TILL ALL MOVED
                             <1>
                             <1> ;-----      FILL IN THE VACATED LINE(S)
                             <1> _R9:                                    ; CLEAR ENTRY
00001F94 88D8                <1>       mov   al, bl              ; attribute to fill with
                             <1> _R10_:
00001F96 E819040000          <1>         CALL    _R18                    ; CLEAR THAT ROW
00001F9B 6681EFB01F          <1>       SUB   DI, 2000h-80        ; POINT TO NEXT LINE
00001FA0 FECF                <1>       dec   bh                  ; number of lines to fill
00001FA2 75F2                <1>         JNZ     short _R10_             ; CLEAR LOOP
00001FA4 C3                  <1>       retn                      ; EVERYYHING DONE
                             <1>
                             <1> _R11:                                   ; BLANK_FIELD
00001FA5 88F7                <1>       mov   bh, dh              ; set blank count to everything in field
00001FA7 EBEB                <1>         JMP     short _R9               ; CLEAR THE FIELD
                             <1>
                             <1> vga_graphics_up:
                             <1>       ; 08/08/2016
                             <1>       ; 07/08/2016
                             <1>       ; 04/08/2016
                             <1>       ; 01/08/2016
                             <1>       ; 31/07/2016
                             <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'biosfn_scroll'
                             <1>       ;
                             <1>
                             <1>       ; cl = upper left column
                             <1>       ; ch = upper left row
                             <1>       ; dl = lower rigth column
                             <1>       ; dh = lower right row
                             <1>       ;
                             <1>       ; al = line count (AL=0 means blank entire fields)
                             <1>       ; bl = fill value for blanked lines
                             <1>       ; bh = unused
                             <1>       ;
                             <1>       ; ah = [CRT_MODE], current video mode
                             <1>
00001FA9 88C7                <1>       mov   bh, al ; 31/07/2016
00001FAB BE[E65E0000]        <1>       mov   esi, vga_g_modes
00001FB0 89F7                <1>       mov   edi, esi
00001FB2 83C708              <1>       add   edi, vga_g_mode_count
                             <1> vga_g_up_0:
00001FB5 AC                  <1>       lodsb
00001FB6 38E0                <1>       cmp   al, ah ; [CRT_MODE]
00001FB8 7405                <1>       je    short vga_g_up_1
00001FBA 39FE                <1>       cmp   esi, edi
00001FBC 72F7                <1>       jb    short vga_g_up_0
                             <1>       ;xor  bh, bh ; 31/07/2016)
00001FBE C3                  <1>       retn  ; nothing to do
                             <1> vga_g_up_1:
00001FBF 88F8                <1>       mov   al, bh ; 31/07/2016
00001FC1 83C64F              <1>       add   esi, vga_g_memmodel - (vga_g_modes + 1)
```

```
                             <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
                             <1>
                             <1>       ; if(rlr>=nbrows)rlr=nbrows-1;
                             <1>       ; if(clr>=nbcols)clr=nbcols-1;
                             <1>       ; if(nblines>nbrows)nblines=0;
                             <1>       ; cols=clr-cul+1;
                             <1>
00001FC4 3A35[CA5E0000]      <1>       cmp    dh, [VGA_ROWS]
00001FCA 7208                <1>       jb     short vga_g_up_2
00001FCC 8A35[CA5E0000]      <1>       mov    dh, [VGA_ROWS]
00001FD2 FECE                <1>       dec    dh
                             <1> vga_g_up_2:
00001FD4 3A15[C45E0000]      <1>       cmp    dl, [CRT_COLS]  ; = [VGA_COLS]
00001FDA 7208                <1>       jb     short vga_g_up_3
00001FDC 8A15[C45E0000]      <1>       mov    dl, [CRT_COLS]
00001FE2 FECA                <1>       dec    dl
                             <1> vga_g_up_3:
00001FE4 3A05[CA5E0000]      <1>       cmp    al, [VGA_ROWS]
00001FEA 7602                <1>       jna    short vga_g_up_4
00001FEC 28C0                <1>       sub    al, al ; 0
                             <1> vga_g_up_4:
00001FEE 88D7                <1>       mov    bh, dl ; clr
00001FF0 28CF                <1>       sub    bh, cl ; cul
00001FF2 FEC7                <1>       inc    bh ; cols = clr-cul+1
                             <1>
00001FF4 20C0                <1>       and    al, al ; nblines = 0
00001FF6 755D                <1>       jnz    short vga_g_up_6
00001FF8 20ED                <1>       and    ch, ch ; rul = 0
00001FFA 7559                <1>       jnz    short vga_g_up_6
00001FFC 20C9                <1>       and    cl, cl ; cul = 0
00001FFE 7555                <1>       jnz    short vga_g_up_6
                             <1>
00002000 6650                <1>       push   ax
00002002 A0[CA5E0000]        <1>       mov    al, [VGA_ROWS]
00002007 FEC8                <1>       dec    al
00002009 38C6                <1>       cmp    dh, al ; rlr = nbrows-1
0000200B 7546                <1>       jne    short vga_g_up_5
0000200D A0[C45E0000]        <1>         mov     al, [CRT_COLS]  ; = VGA_COLS
00002012 FEC8                <1>       dec    al
00002014 38C2                <1>       cmp    dl, al ; clr = nbcols-1
00002016 753B                <1>       jne    short vga_g_up_5
00002018 6658                <1>       pop    ax
                             <1>
0000201A 66B80502            <1>       mov    ax, 0205h
0000201E 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002022 66EF                <1>       out    dx, ax
00002024 A0[CA5E0000]        <1>       mov    al, [VGA_ROWS]
00002029 8A25[C45E0000]      <1>       mov    ah, [CRT_COLS] ; = [VGA_COLS]
0000202F F6E4                <1>       mul    ah
00002031 0FB7D0              <1>       movzx  edx, ax
                             <1>       ; 08/08/2016
00002034 0FB605[C65E0000]    <1>       movzx  eax, byte [CHAR_HEIGHT]
0000203B F7E2                <1>       mul    edx
                             <1>       ; eax = byte count
0000203D 89C1                <1>       mov    ecx, eax
                             <1>       ;; 07/08/2016
                             <1>       ;shl   dx, 3 ; * 8 ; * [CHAR_HEIGHT]
                             <1>       ;mov   ecx, edx
0000203F 88D8                <1>       mov    al, bl ; fill value for blanked lines
00002041 BF00000A00          <1>       mov    edi, 0A0000h
00002046 F3AA                <1>       rep    stosb
                             <1>
00002048 66B80500            <1>       mov    ax, 5
0000204C 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002050 66EF                <1>       out    dx, ax ; 0005h
                             <1>
00002052 C3                  <1>       retn
                             <1>
                             <1> vga_g_up_5:
00002053 6658                <1>       pop    ax
                             <1>
                             <1> vga_g_up_6:
                             <1>       ; [ESI] = VGA memory model number for current video mode
                             <1>         ;
                             <1>         ; LINEAR8 equ 5
                             <1>         ; PLANAR4 equ 4
                             <1>         ; PLANAR1 equ 3
                             <1>
00002055 803E04              <1>       cmp    byte [esi], PLANAR4
00002058 7424                <1>       je     short vga_g_up_planar
0000205A 803E03              <1>       cmp    byte [esi], PLANAR1
0000205D 741F                <1>       je     short vga_g_up_planar
                             <1> vga_g_up_linear8:
                             <1>       ; 07/07/2016 (TEMPORARY)
                             <1>       ;
                             <1>       ; cl = upper left column ; cul
                             <1>       ; ch = upper left row ; rul
                             <1>       ; dl = lower rigth column ; clr
                             <1>       ; dh = lower right row ; rlr
                             <1>
                             <1> vga_g_up_l0:
                             <1>       ;{for(i=rul;i<=rlr;i++)
                             <1>       ; if((i+nblines>rlr)||(nblines==0))
0000205F 08C0                <1>       or     al, al
00002061 7414                <1>       jz     short vga_g_up_l2
00002063 88C4                <1>       mov    ah, al
00002065 00EC                <1>       add    ah, ch ; i+nblines
                             <1>       ;jc    short vga_g_up_l2
00002067 38F4                <1>       cmp    ah, dh
00002069 770C                <1>       ja     short vga_g_up_l2
                             <1>       ; else
                             <1>       ;  vgamem_copy_pl4(cul,i+nblines,i,cols,nbcols,cheight);
0000206B E8F2000000          <1>       call   vgamem_copy_l8
```

```
                             <1> vga_g_up_l1:
00002070 FEC5                <1>       inc    ch
00002072 38F5                <1>       cmp    ch, dh
00002074 76E9                <1>       jna    short vga_g_up_l0
00002076 C3                  <1>       retn
                             <1> vga_g_up_l2:
                             <1>       ; vgamem_fill_pl4(cul,i,cols,nbcols,cheight,attr);
00002077 E850010000          <1>       call   vgamem_fill_l8
0000207C EBF2                <1>       jmp    short vga_g_up_l1
                             <1> 
                             <1> vga_g_up_planar:
                             <1>       ; cl = upper left column ; cul
                             <1>       ; ch = upper left row ; rul
                             <1>       ; dl = lower rigth column ; clr
                             <1>       ; dh = lower right row ; rlr
                             <1> vga_g_up_pl0:
                             <1>       ;{for(i=rul;i<=rlr;i++)
                             <1>       ; if((i+nblines>rlr)||(nblines==0))
0000207E 20C0                <1>       and    al, al
00002080 7414                <1>       jz     short vga_g_up_pl2
00002082 88C4                <1>       mov    ah, al
00002084 00EC                <1>       add    ah, ch ; i+nblines
                             <1>       ;jc    short vga_g_up_pl2
00002086 38F4                <1>       cmp    ah, dh
00002088 770C                <1>       ja     short vga_g_up_pl2
                             <1>       ; else
                             <1>       ;  vgamem_copy_pl4(cul,i+nblines,i,cols,nbcols,cheight);
0000208A E80E000000          <1>       call   vgamem_copy_pl4
                             <1> vga_g_up_pl1:
0000208F FEC5                <1>       inc    ch
00002091 38F5                <1>       cmp    ch, dh
00002093 76E9                <1>       jna    short vga_g_up_pl0
00002095 C3                  <1>       retn
                             <1> vga_g_up_pl2:
                             <1>       ; vgamem_fill_pl4(cul,i,cols,nbcols,cheight,attr);
00002096 E870000000          <1>       call   vgamem_fill_pl4
0000209B EBF2                <1>       jmp    short vga_g_up_pl1
                             <1> 
                             <1> vgamem_copy_pl4:
                             <1>       ; 08/08/2016
                             <1>       ; 07/08/2016
                             <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'vgamem_copy_pl4'
                             <1>       ;
                             <1>       ; vgamem_copy_pl4(xstart,ysrc,ydest,cols,nbcols,cheight)
                             <1>       ; cl = xstart, ah = ysrc (i+nblines), ch = ydest (i),
                             <1>       ; bh = cols, [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight
                             <1> 
                             <1>       ; src=ysrc*cheight*nbcols+xstart;
                             <1>       ; dest=ydest*cheight*nbcols+xstart;
                             <1> 
0000209D 52                  <1>       push   edx
0000209E 50                  <1>       push   eax
                             <1> 
                             <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0105)
0000209F 66B80501            <1>       mov    ax, 0105h
000020A3 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
000020A7 66EF                <1>       out    dx, ax
                             <1> 
                             <1>       ; 07/08/2016
                             <1>       ;mov      ah, [esp+1]
                             <1>       ;movzx edx, ah ; ysrc
000020A9 0FB6542401          <1>       movzx  edx, byte [esp+1]
                             <1>       ; 08/08/2016
000020AE 0FB605[C65E0000]    <1>       movzx  eax, byte [CHAR_HEIGHT]
000020B5 8A25[C45E0000]      <1>       mov    ah, [CRT_COLS] ; nbcols
000020BB F6E4                <1>       mul    ah
                             <1>       ;; 07/08/2016
                             <1>       ;movzx eax, byte [CRT_COLS]
                             <1>       ;shl   ax, 3 ; * 8 ; * [CHAR_HEIGHT]
000020BD 50                  <1>       push   eax ; cheight * nbcols
000020BE F7E2                <1>       mul    edx ; * ysrc
                             <1>       ; eax = ysrc * cheight * nbcols
                             <1>       ; edx = 0
000020C0 88CA                <1>       mov    dl, cl ; edx = xstart
000020C2 01D0                <1>       add    eax, edx
000020C4 89C6                <1>       mov    esi, eax ; src
000020C6 88EA                <1>       mov    dl, ch ; ydest
000020C8 58                  <1>       pop    eax ; cheight * nbcols
000020C9 F7E2                <1>       mul    edx
                             <1>       ; eax = ydest * cheight * nbcols
000020CB 88CA                <1>       mov    dl, cl ; edx = xstart
000020CD 01D0                <1>       add    eax, edx
000020CF 89C7                <1>       mov    edi, eax ; dest
                             <1>       ; esi = src
                             <1>       ; edi = dest
                             <1>       ; for(i=0;i<cheight;i++)
                             <1>       ; {
                             <1>       ;  memcpyb(0xa000,dest+i*nbcols,0xa000,src+i*nbcols,cols);
                             <1>       ; }
000020D1 51                  <1>       push   ecx
000020D2 B900000A00          <1>       mov    ecx, 0A0000h
000020D7 01CE                <1>       add    esi, ecx
000020D9 01CF                <1>       add    edi, ecx
                             <1>       ; 08/08/2016
000020DB 8A35[C65E0000]      <1>       mov    dh, [CHAR_HEIGHT]
                             <1>       ;; 07/08/2016
                             <1>       ;mov   dh, 8 ; 07/08/2016
000020E1 28D2                <1>       sub    dl, dl ; i
```

```
                             <1> vgamem_copy_pl4_0:
000020E3 56                  <1>       push   esi
000020E4 57                  <1>       push   edi
000020E5 0FB605[C45E0000]    <1>       movzx  eax, byte [CRT_COLS]
000020EC F6E2                <1>       mul    dl
                             <1>       ; eax = i * nbcols
000020EE 01C7                <1>       add    edi, eax ; dest+i*nbcols
000020F0 01C6                <1>       add    esi, eax
000020F2 0FB6CF              <1>       movzx  ecx, bh ; cols
000020F5 F3A4                <1>       rep    movsb
000020F7 5F                  <1>       pop    edi
000020F8 5E                  <1>       pop    esi
000020F9 FECE                <1>       dec    dh
000020FB 75E6                <1>       jnz    short vgamem_copy_pl4_0
                             <1> vgamem_copy_pl4_1:
000020FD 59                  <1>       pop    ecx
                             <1>
                             <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0005);
000020FE 66B80500            <1>       mov    ax, 0005h
00002102 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002106 66EF                <1>       out    dx, ax
                             <1>
00002108 58                  <1>       pop    eax
00002109 5A                  <1>       pop    edx
                             <1>
0000210A C3                  <1>       retn
                             <1>
                             <1> vgamem_fill_pl4:
                             <1>       ; 08/08/2016
                             <1>       ; 07/08/2016
                             <1>       ; 04/08/2016
                             <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'vgamem_fill_pl4'
                             <1>       ;
                             <1>       ; vgamem_fill_pl4(xstart,ystart,cols,nbcols,cheight,attr)
                             <1>       ; cl = xstart, edi = ch = ystart, bh = cols,
                             <1>       ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight, attr = 0
                             <1>
                             <1>       ; dest=ystart*cheight*nbcols+xstart;
0000210B 52                  <1>       push   edx
0000210C 50                  <1>       push   eax
                             <1>
                             <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0205)
0000210D 66B80502            <1>       mov    ax, 0205h
00002111 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002115 66EF                <1>       out    dx, ax
                             <1>
                             <1>              ; 08/08/2016
00002117 0FB605[C65E0000]    <1>       movzx   eax, byte [CHAR_HEIGHT]
0000211E F6E5                <1>       mul    ch
                             <1>       ;; 07/08/2016
                             <1>       ;movzx eax, ch
                             <1>       ;shl   ax, 3 ; * 8 ; * [CHAR_HEIGHT]
00002120 0FB615[C45E0000]    <1>       movzx  edx, byte [CRT_COLS] ; = [VGA_COLS]
00002127 F7E2                <1>       mul    edx
                             <1>       ; edx  = 0
00002129 88CA                <1>       mov    dl, cl
0000212B 01D0                <1>       add    eax, edx
0000212D 89C7                <1>       mov    edi, eax
                             <1>       ; edi = dest
                             <1>       ; for(i=0;i<cheight;i++)
                             <1>       ; {
                             <1>       ;  memsetb(0xa000,dest+i*nbcols,attr,cols);
                             <1>       ; }
0000212F 81C700000A00        <1>       add    edi, 0A0000h
00002135 51                  <1>       push   ecx
                             <1>       ; 08/08/2016
00002136 8A35[C65E0000]      <1>       mov    dh, [CHAR_HEIGHT]
                             <1>       ;; 07/08/2016
                             <1>       ;mov   dh, 8 ; 07/08/2016
0000213C 28D2                <1>       sub    dl, dl ; i
                             <1> vgamem_fill_pl4_0:
0000213E 57                  <1>       push   edi
0000213F 0FB605[C45E0000]    <1>       movzx  eax, byte [CRT_COLS]
00002146 F6E2                <1>       mul    dl
                             <1>       ; eax = i * nbcols
00002148 01C7                <1>       add    edi, eax ; dest+i*nbcols
0000214A 88D8                <1>       mov    al, bl ; attr ; 04/08/2016
0000214C 0FB6CF              <1>       movzx  ecx, bh ; cols
0000214F F3AA                <1>       rep    stosb
00002151 5F                  <1>       pop    edi
00002152 75EA                <1>       jnz    short vgamem_fill_pl4_0
                             <1> vgamem_fill_pl4_1:
00002154 59                  <1>       pop    ecx
                             <1>
                             <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0005);
00002155 66B80500            <1>       mov    ax, 0005h
00002159 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
0000215D 66EF                <1>       out    dx, ax
                             <1>
0000215F 58                  <1>       pop    eax
00002160 5A                  <1>       pop    edx
                             <1>
00002161 C3                  <1>       retn
                             <1>
                             <1> vgamem_copy_l8:
                             <1>       ; 08/08/2016
                             <1>       ; 07/08/2016
                             <1>       ; 06/08/2016
```

```
                              <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
                              <1>       ;
                              <1>       ; TEMPORARY
                              <1>       ;
                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
                              <1>       ; vgabios-0.7a (2011)
                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
                              <1>       ; 'vgabios.c', 'vgamem_copy_pl4'
                              <1>       ;
                              <1>       ; vgamem_copy_pl4(xstart,ysrc,ydest,cols,nbcols,cheight)
                              <1>       ; cl = xstart, ah = ysrc (i+nblines), ch = ydest (i),
                              <1>       ; bh = cols, [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight
                              <1>
                              <1>       ; src=ysrc*cheight*nbcols+xstart;
                              <1>       ; dest=ydest*cheight*nbcols+xstart;
                              <1>
00002162 52                   <1>       push   edx
00002163 50                   <1>       push   eax
                              <1>
                              <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0105)
                              <1>       ;mov   ax, 0105h
                              <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
                              <1>       ;out   dx, ax
                              <1>
                              <1>       ;mov   ah, [esp+1]
                              <1>
00002164 0FB6D4               <1>       movzx  edx, ah ; ysrc
                              <1>       ; 08/08/2016
00002167 0FB605[C65E0000]     <1>       movzx  eax, byte [CHAR_HEIGHT]
0000216E 8A25[C45E0000]       <1>       mov    ah, [CRT_COLS] ; nbcols
00002174 F6E4                 <1>       mul    ah
                              <1>       ;; 07/08/2016
                              <1>       ;movzx eax, byte [CRT_COLS]
                              <1>       ;shl   ax, 3 ; * 8 ; * [CHAR_HEIGHT]
00002176 50                   <1>       push   eax ; cheight * nbcols
00002177 F7E2                 <1>       mul    edx ; * ysrc
                              <1>       ; eax = ysrc * cheight * nbcols
                              <1>       ; edx = 0
00002179 88CA                 <1>       mov    dl, cl ; edx = xstart
0000217B 01D0                 <1>       add    eax, edx
0000217D 89C6                 <1>       mov    esi, eax ; src
0000217F 66C1E603             <1>       shl    si, 3 ; * 8 ; 06/08/2016
00002183 88EA                 <1>       mov    dl, ch ; ydest
00002185 58                   <1>       pop    eax ; cheight * nbcols
00002186 F7E2                 <1>       mul    edx
                              <1>       ; eax = ydest * cheight * nbcols
00002188 88CA                 <1>       mov    dl, cl ; edx = xstart
0000218A 01D0                 <1>       add    eax, edx
0000218C 89C7                 <1>       mov    edi, eax ; dest
0000218E 66C1E703             <1>       shl    di, 3 ; * 8 ; 06/08/2016
                              <1>       ; esi = src
                              <1>       ; edi = dest
                              <1>       ; for(i=0;i<cheight;i++)
                              <1>       ; {
                              <1>       ;  memcpyb(0xa000,dest+i*nbcols,0xa000,src+i*nbcols,cols);
                              <1>       ; }
00002192 51                   <1>       push   ecx
00002193 B900000A00           <1>       mov    ecx, 0A0000h
00002198 01CE                 <1>       add    esi, ecx
0000219A 01CF                 <1>       add    edi, ecx
                              <1>       ; 08/08/2016
0000219C 8A35[C65E0000]       <1>       mov    dh, [CHAR_HEIGHT]
                              <1>       ;; 07/08/2016
                              <1>       ;mov   dh, 8 ; 07/08/2016
000021A2 28D2                 <1>       sub    dl, dl ; i
                              <1> vgamem_copy_l8_0:
000021A4 56                   <1>       push   esi
000021A5 57                   <1>       push   edi
000021A6 0FB605[C45E0000]     <1>       movzx  eax, byte [CRT_COLS]
000021AD F6E2                 <1>       mul    dl
                              <1>       ; eax = i * nbcols
000021AF 66C1E003             <1>       shl    ax, 3 ; * 8 ; 06/08/2016
000021B3 01C7                 <1>       add    edi, eax ; dest+i*nbcols
000021B5 01C6                 <1>       add    esi, eax
000021B7 0FB6CF               <1>       movzx  ecx, bh ; cols
000021BA 66C1E103             <1>       shl    cx, 3 ; * 8 ; 06/08/2016
000021BE F3A4                 <1>       rep    movsb
000021C0 5F                   <1>       pop    edi
000021C1 5E                   <1>       pop    esi
000021C2 FEC2                 <1>       inc    dl ; 06/08/2016
000021C4 FECE                 <1>       dec    dh
000021C6 75DC                 <1>       jnz    short vgamem_copy_l8_0
                              <1> vgamem_copy_l8_1:
000021C8 59                   <1>       pop    ecx
                              <1>
                              <1>       ;; outw(VGAREG_GRDC_ADDRESS, 0x0005);
                              <1>       ;mov   ax, 0005h
                              <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
                              <1>       ;out   dx, ax
                              <1>
000021C9 58                   <1>       pop    eax
000021CA 5A                   <1>       pop    edx
                              <1>
000021CB C3                   <1>       retn
                              <1>
                              <1> vgamem_fill_l8:
                              <1>       ; 08/08/2016
                              <1>       ; 07/08/2016
                              <1>       ; 06/08/2016
                              <1>       ; 04/08/2016
                              <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
                              <1>       ;
                              <1>       ; TEMPORARY
```

```
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'vgamem_fill_pl4'
                             <1>       ;
                             <1>       ; vgamem_fill_pl4(xstart,ystart,cols,nbcols,cheight,attr)
                             <1>       ; cl = xstart, edi = ch = ystart, bh = cols,
                             <1>       ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight, attr = 0
                             <1>
                             <1>       ; dest=ystart*cheight*nbcols+xstart;
000021CC 52                  <1>       push   edx
000021CD 50                  <1>       push   eax
                             <1>
                             <1>       ;; outw(VGAREG_GRDC_ADDRESS, 0x0205)
                             <1>       ;mov   ax, 0205h
                             <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
                             <1>       ;out   dx, ax
                             <1>
                             <1>         ; 08/08/2016
000021CE 0FB605[C65E0000]    <1>       movzx   eax, byte [CHAR_HEIGHT]
000021D5 F6E5                <1>       mul    ch
                             <1>       ;; 07/08/2016
                             <1>       ;movzx eax, ch
                             <1>       ;shl   ax, 3 ; * 8 ; * [CHAR_HEIGHT]
000021D7 0FB615[C45E0000]    <1>       movzx  edx, byte [CRT_COLS] ; = [VGA_COLS]
000021DE F7E2                <1>       mul    edx
                             <1>       ; edx  = 0
000021E0 88CA                <1>       mov    dl, cl
000021E2 01D0                <1>       add    eax, edx
000021E4 89C7                <1>       mov    edi, eax
000021E6 66C1E703            <1>       shl    di, 3 ; * 8 ; 06/08/2016
                             <1>       ; edi = dest
                             <1>       ; for(i=0;i<cheight;i++)
                             <1>       ; {
                             <1>       ;  memsetb(0xa000,dest+i*nbcols,attr,cols);
                             <1>       ; }
000021EA 81C700000A00        <1>       add    edi, 0A0000h
000021F0 51                  <1>       push   ecx
                             <1>       ; 08/08/2016
000021F1 8A35[C65E0000]      <1>       mov    dh, [CHAR_HEIGHT]
                             <1>       ;; 07/08/2016
                             <1>       ;mov   dh, 8 ; 07/08/2016
000021F7 28D2                <1>       sub    dl, dl ; i
                             <1> vgamem_fill_l8_0:
000021F9 57                  <1>       push   edi
000021FA 0FB605[C45E0000]    <1>       movzx  eax, byte [CRT_COLS]
00002201 F6E2                <1>       mul    dl
                             <1>       ; eax = i * nbcols
00002203 66C1E003            <1>       shl    ax, 3 ; * 8 ; 06/08/2016
00002207 01C7                <1>       add    edi, eax ; dest+i*nbcols
00002209 88D8                <1>       mov    al, bl ; attr ; 04/08/2016
0000220B 0FB6CF              <1>       movzx  ecx, bh ; cols
0000220E 66C1E103            <1>       shl    cx, 3 ; * 8 ; 06/08/2016
00002212 F3AA                <1>       rep    stosb
00002214 5F                  <1>       pop    edi
00002215 FEC2                <1>       inc    dl ; 06/08/2016
00002217 FECE                <1>       dec    dh
00002219 75DE                <1>       jnz    short vgamem_fill_l8_0
                             <1> vgamem_fill_l8_1:
0000221B 59                  <1>       pop    ecx
                             <1>
                             <1>       ;; outw(VGAREG_GRDC_ADDRESS, 0x0005);
                             <1>       ;mov   ax, 0005h
                             <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
                             <1>       ;out   dx, ax
                             <1>
0000221C 58                  <1>       pop    eax
0000221D 5A                  <1>       pop    edx
                             <1>
0000221E C3                  <1>       retn
                             <1>
                             <1> vga_graphics_down:
                             <1>       ; 08/08/2016
                             <1>       ; 07/08/2016
                             <1>       ; 31/07/2016
                             <1>       ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'biosfn_scroll'
                             <1>       ;
                             <1>
                             <1>       ; cl = upper left column
                             <1>       ; ch = upper left row
                             <1>       ; dl = lower rigth column
                             <1>       ; dh = lower right row
                             <1>       ;
                             <1>       ; al = line count (AL=0 means blank entire fields)
                             <1>       ; bl = fill value for blanked lines
                             <1>       ; bh = unused
                             <1>       ;
                             <1>       ; ah = [CRT_MODE], current video mode
                             <1>
0000221F FC                  <1>         cld     ; !!! Clear direction flag !!!
                             <1>
00002220 88C7                <1>       mov    bh, al ; 31/07/2016
                             <1>
00002222 BE[DE5E0000]        <1>       mov    esi, vga_modes
00002227 89F7                <1>       mov    edi, esi
00002229 83C710              <1>       add    edi, vga_mode_count
                             <1> vga_g_down_0:
```

```
0000222C AC                  <1>       lodsb
0000222D 38E0                <1>       cmp    al, ah ; [CRT_MODE]
0000222F 7405                <1>       je     short vga_g_down_1
00002231 39FE                <1>       cmp    esi, edi
00002233 72F7                <1>       jb     short vga_g_down_0
                             <1>       ; xor  bh, bh ; 31/07/2016
00002235 C3                  <1>       retn   ; nothing to do
                             <1> vga_g_down_1:
00002236 88F8                <1>       mov    al, bh ; 31/07/2016
00002238 83C64F              <1>       add    esi, vga_memmodel - (vga_modes + 1)
                             <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
                             <1>
                             <1>       ; if(rlr>=nbrows)rlr=nbrows-1;
                             <1>       ; if(clr>=nbcols)clr=nbcols-1;
                             <1>       ; if(nblines>nbrows)nblines=0;
                             <1>       ; cols=clr-cul+1;
                             <1>
0000223B 3A35[CA5E0000]      <1>       cmp    dh, [VGA_ROWS]
00002241 7208                <1>       jb     short vga_g_down_2
00002243 8A35[CA5E0000]      <1>       mov    dh, [VGA_ROWS]
00002249 FECE                <1>       dec    dh
                             <1> vga_g_down_2:
0000224B 3A15[C45E0000]      <1>       cmp    dl, [CRT_COLS]  ; = [VGA_COLS]
00002251 7208                <1>       jb     short vga_g_down_3
00002253 8A15[C45E0000]      <1>       mov    dl, [CRT_COLS]
00002259 FECA                <1>       dec    dl
                             <1> vga_g_down_3:
0000225B 3A05[CA5E0000]      <1>       cmp    al, [VGA_ROWS]
00002261 7602                <1>       jna    short vga_g_down_4
00002263 28C0                <1>       sub    al, al ; 0
                             <1> vga_g_down_4:
00002265 88F7                <1>       mov    bh, dh ; clr
00002267 28CF                <1>       sub    bh, cl ; cul
00002269 FEC7                <1>       inc    bh ; cols = clr-cul+1
                             <1>
0000226B 20C0                <1>       and    al, al ; nblines = 0
0000226D 755B                <1>       jnz    short vga_g_down_6
0000226F 20ED                <1>       and    ch, ch ; rul = 0
00002271 7557                <1>       jnz    short vga_g_down_6
00002273 20C9                <1>       and    cl, cl ; cul = 0
00002275 7553                <1>       jnz    short vga_g_down_6
                             <1>
00002277 6650                <1>       push   ax
00002279 A0[CA5E0000]        <1>       mov    al, [VGA_ROWS]
0000227E FEC8                <1>       dec    al
00002280 38C6                <1>       cmp    dh, al ; rlr = nbrows-1
00002282 7544                <1>       jne    short vga_g_down_5
00002284 A0[C45E0000]        <1>         mov     al, [CRT_COLS]  ; = VGA_COLS
00002289 FEC8                <1>       dec    al
0000228B 38C2                <1>       cmp    dl, al ; clr = nbcols-1
0000228D 7539                <1>       jne    short vga_g_down_5
0000228F 6658                <1>       pop    ax
                             <1>
00002291 66B80502            <1>       mov    ax, 0205h
00002295 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002299 66EF                <1>       out    dx, ax
0000229B A0[CA5E0000]        <1>       mov    al, [VGA_ROWS]
000022A0 8A25[C45E0000]      <1>       mov    ah, [CRT_COLS] ; = [VGA_COLS]
000022A6 F6E4                <1>       mul    ah
000022A8 0FB7D0              <1>       movzx  edx, ax
                             <1>       ; 08/08/2016
000022AB 0FB605[C65E0000]    <1>       movzx  eax, byte [CHAR_HEIGHT]
000022B2 F7E2                <1>       mul    edx
                             <1>       ; eax = byte count
000022B4 89C1                <1>       mov    ecx, eax
                             <1>       ;; 07/08/2016
                             <1>       ;shl   dx, 3 ; * 8 ; * [CHAR_HEIGHT]
                             <1>       ;mov   ecx, edx
000022B6 88D8                <1>       mov    al, bl ; fill value for blanked lines
000022B8 BF00000A00          <1>       mov    edi, 0A0000h
000022BD F3AA                <1>       rep    stosb
                             <1>
000022BF B005                <1>       mov    al, 5
000022C1 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
000022C5 66EF                <1>       out    dx, ax ; 0005h
                             <1>
000022C7 C3                  <1>       retn
                             <1>
                             <1> vga_g_down_5:
000022C8 6658                <1>       pop    ax
                             <1>
                             <1> vga_g_down_6:
                             <1>       ; [ESI] = VGA memory model number for current video mode
                             <1>         ;
                             <1>         ; LINEAR8 equ 5
                             <1>         ; PLANAR4 equ 4
                             <1>         ; PLANAR1 equ 3
                             <1>
000022CA 803E04              <1>       cmp    byte [esi], PLANAR4
000022CD 742C                <1>       je     short vga_g_down_planar
000022CF 803E03              <1>       cmp    byte [esi], PLANAR1
000022D2 7427                <1>       je     short vga_g_down_planar
                             <1> vga_g_down_linear8:
                             <1>       ; 07/07/2016 (TEMPORARY)
                             <1>       ;
                             <1>       ; cl = upper left column ; cul
                             <1>       ; ch = upper left row ; rul
                             <1>       ; dl = lower rigth column ; clr
                             <1>       ; dh = lower right row ; rlr
                             <1>
                             <1> vga_g_down_l0:
                             <1>       ;{for(i=rlr;i>=rul;i--)
                             <1>         ; if((i<rul+nblines)||(nblines==0))
```

```
000022D4 08C0                <1>       or     al, al
000022D6 741C                <1>       jz     short vga_g_down_l2
000022D8 88C4                <1>       mov    ah, al
000022DA 00EC                <1>       add    ah, ch
                             <1>       ;jc    short vga_g_down_l2
000022DC 86EE                <1>       xchg   ch, dh
000022DE 38E5                <1>       cmp    ch, ah
000022E0 7212                <1>       jb     short vga_g_down_l2
000022E2 88EC                <1>       mov    ah, ch
000022E4 28C4                <1>       sub    ah, al ; ah = i - nblines
                             <1>       ; else
                             <1>       ; vgamem_copy_pl4(cul,i,i-nblines,cols,nbcols,cheight);
000022E6 E877FEFFFF          <1>       call   vgamem_copy_l8
                             <1> vga_g_down_l1:
000022EB 86F5                <1>       xchg   dh, ch
000022ED FECE                <1>       dec    dh
000022EF 38EE                <1>       cmp    dh, ch
000022F1 73E1                <1>       jnb    short vga_g_down_l0
000022F3 C3                  <1>       retn
                             <1>
                             <1> vga_g_down_l2:
                             <1>       ; vgamem_fill_pl4(cul,i,cols,nbcols,cheight,attr);
000022F4 E8D3FEFFFF          <1>       call   vgamem_fill_l8
000022F9 EBF0                <1>       jmp    short vga_g_down_l1
                             <1>
                             <1> vga_g_down_planar:
                             <1>       ; cl = upper left column ; cul
                             <1>       ; ch = upper left row ; rul
                             <1>       ; dl = lower rigth column ; clr
                             <1>       ; dh = lower right row ; rlr
                             <1> vga_g_down_pl0:
                             <1>       ;{for(i=rlr;i>=rul;i--)
                             <1>         ; if((i<rul+nblines)||(nblines==0))
000022FB 08C0                <1>       or     al, al
000022FD 741C                <1>       jz     short vga_g_down_pl2
000022FF 88C4                <1>       mov    ah, al
00002301 00EC                <1>       add    ah, ch
                             <1>       ;jc    short vga_g_down_pl2
00002303 86EE                <1>       xchg   ch, dh
00002305 38E5                <1>       cmp    ch, ah
00002307 7212                <1>       jb     short vga_g_down_pl2
00002309 88EC                <1>       mov    ah, ch
0000230B 28C4                <1>       sub    ah, al ; ah = i - nblines
                             <1>       ; else
                             <1>       ; vgamem_copy_pl4(cul,i,i-nblines,cols,nbcols,cheight);
0000230D E88BFDFFFF          <1>       call   vgamem_copy_pl4
                             <1> vga_g_down_pl1:
00002312 86F5                <1>       xchg   dh, ch
00002314 FECE                <1>       dec    dh
00002316 38EE                <1>       cmp    dh, ch
00002318 73E1                <1>       jnb    short vga_g_down_pl0
0000231A C3                  <1>       retn
                             <1>
                             <1> vga_g_down_pl2:
                             <1>       ; vgamem_fill_pl4(cul,i,cols,nbcols,cheight,attr);
0000231B E8EBFDFFFF          <1>       call   vgamem_fill_pl4
00002320 EBF0                <1>       jmp    short vga_g_down_pl1
                             <1>
                             <1> ; 07/07/2016
                             <1> ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                             <1> ;-----------------------------------------------------
                             <1> ; SCROLL DOWN
                             <1> ;  THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
                             <1> ; ENTRY --
                             <1> ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                             <1> ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                             <1> ;   BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                             <1> ;  BH = FILL VALUE FOR BLANKED LINES
                             <1> ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                             <1> ;  DS = DATA SEGMENT
                             <1> ;  ES = REGEN SEGMENT
                             <1> ; EXIT --
                             <1> ;  NOTHING, THE SCREEN IS SCROLLED
                             <1> ;-------------------------------------------------------
                             <1>
                             <1>       ; cl = upper left column
                             <1>       ; ch = upper left row
                             <1>       ; dl = lower rigth column
                             <1>       ; dh = lower right row
                             <1>       ;
                             <1>       ; al = line count (AL=0 means blank entire fields)
                             <1>       ; bl = fill value for blanked lines
                             <1>       ; bh = unused
                             <1>
                             <1> GRAPHICS_DOWN:
                             <1>       ; 07/07/2016
                             <1>       ;AH = Current video mode, [CRT_MODE]
                             <1>       ;STD                      ; SET DIRECTION
00002322 80FC07              <1>       cmp    ah, 7
00002325 0F87F4FEFFFF        <1>         ja      vga_graphics_down
                             <1>       ;je    _n0
                             <1>
0000232B 88C7                <1>       MOV    bh, al             ; save line count in BH
0000232D 6689D0              <1>       MOV    AX, DX             ; GET LOWER RIGHT POSITION INTO AX REG
                             <1>
                             <1> ;-----       USE CHARACTER SUBROUTINE FOR POSITIONING
                             <1> ;-----       ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
                             <1>
00002330 E8F1010000          <1>       CALL   GRAPH_POSN
00002335 0FB7F8              <1>       MOVzx eDI, AX                    ; SAVE RESULT AS DESTINATION ADDRESS
                             <1>
                             <1> ;-----       DETERMINE SIZE OF WINDOW
                             <1>
00002338 6629CA              <1>       SUB    DX, CX
0000233B 6681C20101          <1>         ADD     DX, 101h                 ; ADJUST VALUES
00002340 C0E602              <1>       SAL    DH, 2               ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
                             <1>                                  ; AND EVEN/ODD ROWS
                             <1>
                             <1> ;-----      DETERMINE CRT MODE
                             <1>
00002343 803D[C25E0000]06    <1>       CMP    byte [CRT_MODE], 6  ; TEST FOR MEDIUM RES
0000234A 7307                <1>         JNC     short _R12               ; FIND_SOURCE_DOWN
                             <1>
                             <1> ;-----      MEDIUM RES DOWN
0000234C D0E2                <1>       SAL    DL, 1               ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
0000234E 66D1E7              <1>       SAL    DI, 1               ; OFFSET *2 SINCE 2 BYTES/CHAR
00002351 6647                <1>       INC    DI                  ; POINT TO LAST BYTE
                             <1>
                             <1> ;-----      DETERMINE THE SOURCE ADDRESS IN THE BUFFER
                             <1>
                             <1> _R12:                                    ; FIND_SOURCE_DOWN
00002353 81C700800B00        <1>       add    edi, 0B8000h
00002359 6681C7F000          <1>       ADD    DI, 240                   ; POINT TO LAST ROW OF PIXELS
0000235E C0E702              <1>       sal    bh, 2               ; multiply number of lines by 4
00002361 74(06)              <1>       JZ     short 6                   ; IF ZERO, THEN BLANK ENTIRE FIELD
00002363 B050                <1>       MOV    AL, 80              ; 80 BYTES/ROW
00002365 F6E7                <1>       mul    bh                  ; determine offset to source
00002367 89FE                <1>       MOV    eSI, eDI            ; SET UP SOURCE
00002369 6629C6              <1>       SUB    SI, AX              ; SUBTRACT THE OFFSET
0000236C 88F4                <1>       MOV    AH, DH              ; NUMBER OF ROWS IN FIELD
0000236E 28FC                <1>       sub    ah, bh              ; determine number to move
                             <1>
                             <1> ;-----      LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
                             <1>
                             <1> _R13:                                    ; ROW_LOOP_DOWN
00002370 E823000000          <1>         CALL    _R17                     ; MOVE ONE ROW
00002375 6681EE5020          <1>       SUB    SI, 2000h+80        ; MOVE TO NEXT ROW
0000237A 6681EF5020          <1>       SUB    DI, 2000h+80
0000237F FECC                <1>       DEC    AH                  ; NUMBER OF ROWS TO MOVE
00002381 75ED                <1>         JNZ     short _R13               ; CONTINUE TILL ALL MOVED
                             <1>
                             <1> ;-----      FILL IN THE VACATED LINE(S)
                             <1> _R14:                                    ; CLEAR_ENTRY_DOWN
00002383 88D8                <1>       mov    al, bl              ; attribute to fill with
                             <1> _R15_:                                   ; CLEAR_LOOP_DOWN
00002385 E82A000000          <1>       CALL   _R18                ; CLEAR A ROW
0000238A 6681EF5020          <1>       SUB    DI, 2000h+80        ; POINT TO NEXT LINE
0000238F FECF                <1>       dec    bh                  ; number of lines to fill
00002391 75F2                <1>         JNZ     short _R15_              ; CLEAR_LOOP_DOWN
                             <1>
00002393 C3                  <1>       retn                       ; EVERYYHING DONE
                             <1>
                             <1> _R16:                                    ; BLANK_FIELD_DOWN
00002394 88F7                <1>       mov    bh, dh              ; set blank count to everything in field
00002396 EBEB                <1>         JMP     short _R14               ; CLEAR THE FIELD
                             <1>
                             <1> ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                             <1>
                             <1> ;-----      ROUTINE TO MOVE ONE ROW OF INFORMATION
                             <1>
                             <1> _R17:
00002398 0FB6CA              <1>       MOVzx  ecx, DL                   ; NUMBER OF BYTES IN THE ROW
0000239B 56                  <1>       PUSH   eSI
0000239C 57                  <1>       PUSH   eDI                 ; SAVE POINTERS
0000239D F3A4                <1>       REP    MOVSB               ; MOVE THE EVEN FIELD
0000239F 5F                  <1>       POP    eDI
000023A0 5E                  <1>       POP    eSI
000023A1 6681C60020          <1>       ADD    SI, 2000h
000023A6 6681C70020          <1>       ADD    DI, 2000h           ; POINT TO THE ODD FIELD
000023AB 56                  <1>       PUSH   eSI
000023AC 57                  <1>       PUSH   eDI                 ; SAVE THE POINTERS
000023AD 88D1                <1>       MOV    CL, DL              ; COUNT BACK
000023AF F3A4                <1>       REP    MOVSB               ; MOVE THE ODD FIELD
000023B1 5F                  <1>       POP    eDI
000023B2 5E                  <1>       POP    eSI                 ; POINTERS BACK
000023B3 C3                  <1>       RETn                       ; RETURN TO CALLER
                             <1>
                             <1> ;-----      CLEAR A SINGLE ROW
                             <1>
                             <1> _R18:
000023B4 0FB6CA              <1>       MOVzx  ecx, DL                   ; NUMBER OF BYTES IN FIELD
000023B7 57                  <1>       PUSH   eDI                 ; SAVE POINTER
000023B8 F3AA                <1>       REP    STOSB               ; STORE THE NEW VALUE
000023BA 5F                  <1>       POP    eDI                 ; POINTER BACK
000023BB 6681C70020          <1>       ADD    DI, 2000h           ; POINT TO ODD FIELD
000023C0 57                  <1>       PUSH   eDI
000023C1 88D1                <1>       MOV    CL, DL
000023C3 F3AA                <1>       REP    STOSB               ; FILL THE ODD FIELD
000023C5 5F                  <1>       POP    eDI
000023C6 C3                  <1>       RETn                       ; RETURN TO CALLER
                             <1>
                             <1> ; 04/07/2016
                             <1> ; 01/07/2016
                             <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                             <1> ;--------------------------------------------------
                             <1> ; GRAPHICS WRITE
                             <1> ;  THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
                             <1> ;  POSITION ON THE SCREEN.
                             <1> ; ENTRY --
                             <1> ;  AL = CHARACTER TO WRITE
                             <1> ;  BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
                             <1> ;     IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
                             <1> ;     (0 IS USED FOR THE BACKGROUND COLOR)
```

```
                              <1> ;  CX = NUMBER OF CHARS TO WRITE
                              <1> ;  DS = DATA SEGMENT
                              <1> ;  ES = REGEN SEGMENT
                              <1> ; EXIT --
                              <1> ;  NOTHING IS RETURNED
                              <1> ;
                              <1> ; GRAPHICS READ
                              <1> ;  THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
                              <1> ;  POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO THE
                              <1> ;  CHARACTER GENERATOR CODE POINTS
                              <1> ; ENTRY --
                              <1> ;  NONE (0 IS ASSUMED AS THE BACKGROUND COLOR)
                              <1> ; EXIT --
                              <1> ;  AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
                              <1> ;
                              <1> ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN ROM
                              <1> ;  FOR THE 1ST 128 CHARS.  TO ACCESS CHARS IN THE SECOND HALF, THE USER
                              <1> ;  MUST INITIALIZE THE VECTOR AT INTERRUPT 1FH (LOCATION 0007CH) TO
                              <1> ;  POINT TO THE USER SUPPLIED TABLE OF GRAPHIC IMAGES (8X8 BOXES).
                              <1> ;  FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS
                              <1> ;----------------------------------------------------
                              <1>
                              <1> GRAPHICS_WRITE:
000023C7 25FF000000           <1>       and   eax, 0FFh           ; ZERO TO HIGH OF CODE POINT
000023CC 50                   <1>       PUSH  eAX                 ; SAVE CODE POINT VALUE
                              <1>
                              <1> ;-----      DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
                              <1>
000023CD E84D010000           <1>       CALL  S26                 ; FIND LOCATION IN REGEN BUFFER
000023D2 89C7                 <1>       MOV   eDI, eAX            ; REGEN POINTER IN DI
                              <1>
                              <1> ;-----      DETERMINE REGION TO GET CODE POINTS FROM
                              <1>
000023D4 58                   <1>       POP   eAX                 ; RECOVER CODE POINT
                              <1>
000023D5 BE[B82C0100]         <1>       MOV   eSI, CRT_CHAR_GEN   ; OFFSET OF IMAGES
                              <1>
                              <1> ;-----      DETERMINE GRAPHICS MODE IN OPERATION
                              <1>                                 ; DETERMINE_MODE
000023DA 66C1E003             <1>       SAL   AX, 3               ; MULTIPLY CODE POINT VALUE BY 8
000023DE 01C6                 <1>       ADD   eSI, eAX            ; SI HAS OFFSET OF DESIRED CODES
                              <1>
000023E0 803D[C25E0000]06     <1>       CMP   byte [CRT_MODE], 6
000023E7 7231                 <1>       JC    short S6            ; TEST FOR MEDIUM RESOLUTION MODE
                              <1>
                              <1> ;-----      HIGH RESOLUTION MODE
                              <1>
000023E9 81C700800B00         <1>       add   edi, 0B8000h
                              <1> S1:                             ; HIGH_CHAR
000023EF 57                   <1>       PUSH  eDI                 ; SAVE REGEN POINTER
000023F0 56                   <1>       PUSH  eSI                 ; SAVE CODE POINTER
000023F1 B604                 <1>       MOV   DH, 4               ; NUMBER OF TIMES THROUGH LOOP
                              <1> S2:
000023F3 AC                   <1>       LODSB                     ; GET BYTE FROM CODE POINTS
000023F4 F6C380               <1>       TEST  BL, 80H                    ; SHOULD WE USE THE FUNCTION
000023F7 7515                 <1>       JNZ   short S5            ; TO PUT CHAR IN
000023F9 AA                   <1>       STOSB                     ; STORE IN REGEN BUFFER
000023FA AC                   <1>       LODSB
                              <1> S4:
000023FB 8887FF1F0000         <1>       MOV   [eDI+2000H-1], AL ; STORE IN SECOND HALF
00002401 83C74F               <1>       ADD   eDI, 79                    ; MOVE TO NEXT ROW IN REGEN
00002404 FECE                 <1>       DEC   DH                  ; DONE WITH LOOP
00002406 75EB                 <1>       JNZ   short S2
00002408 5E                   <1>       POP   eSI
00002409 5F                   <1>       POP   eDI                 ; RECOVER REGEN POINTER
0000240A 47                   <1>       INC   eDI                 ; POINT TO NEXT CHAR POSITION
0000240B E2E2                 <1>       LOOP  S1                  ; MORE CHARS TO WRITE
0000240D C3                   <1>       retn
                              <1>
                              <1> S5:
0000240E 3207                 <1>       XOR   AL, [eDI]           ; EXCLUSIVE OR WITH CURRENT
00002410 AA                   <1>       STOSB                     ; STORE THE CODE POINT
00002411 AC                   <1>       LODSB                     ; AGAIN FOR ODD FIELD
00002412 3287FF1F0000         <1>       XOR   AL, [eDI+2000H-1]
00002418 EBE1                 <1>       JMP   short S4            ; BACK TO MAINSTREAM
                              <1>
                              <1> ;-----      MEDIUM RESOLUTION WRITE
                              <1> S6:                             ; MED_RES_WRITE
0000241A 88DA                 <1>       MOV   DL, BL              ; SAVE HIGH COLOR BIT
0000241C 66D1E7               <1>       SAL   DI, 1               ; OFFSET*2 SINCE 2 BYTES/CHAR
                              <1>                                 ; EXPAND BL TO FULL WORD OF COLOR
0000241F 80E303               <1>       AND   BL, 3               ; ISOLATE THE COLOR BITS ( LOW 2 BITS )
00002422 B055                 <1>       MOV   AL, 055H            ; GET BIT CONVERSION MULTIPLIER
00002424 F6E3                 <1>       MUL   BL                  ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS
00002426 88C3                 <1>       MOV   BL, AL              ; PLACE BACK IN WORK REGISTER
00002428 88C7                 <1>       MOV   BH, AL              ; EXPAND TO 8 REPLICATIONS OF COLOR BITS
0000242A 81C700800B00         <1>       add   edi, 0B8000h
                              <1> S7:                                     ; MED_CHAR
00002430 57                   <1>       PUSH  eDI                 ; SAVE REGEN POINTER
00002431 56                   <1>       PUSH  eSI                 ; SAVE THE CODE POINTER
00002432 B604                 <1>       MOV   DH, 4               ; NUMBER OF LOOPS
                              <1> S8:
00002434 AC                   <1>       LODSB                     ; GET CODE POINT
00002435 E8B3000000           <1>       CALL  S21                 ; DOUBLE UP ALL THE BITS
0000243A 6621D8               <1>       AND   AX, BX              ; CONVERT TO FOREGROUND COLOR ( 0 BACK )
0000243D 86E0                 <1>       XCHG  AH, AL              ; SWAP HIGH/LOW BYTES FOR WORD MOVE
0000243F F6C280               <1>       TEST  DL, 80H                    ; IS THIS XOR FUNCTION
00002442 7403                 <1>       JZ    short S9            ; NO, STORE IT IN AS IS
00002444 663307               <1>       XOR   AX, [eDI]           ; DO FUNCTION WITH LOW/HIGH
                              <1> S9:
00002447 668907               <1>       MOV   [eDI], AX           ; STORE FIRST BYTE HIGH, SECOND LOW
0000244A AC                   <1>       LODSB                     ; GET CODE POINT
0000244B E89D000000           <1>       CALL  S21
```

```
00002450 6621D8             <1>       AND   AX, BX             ; CONVERT TO COLOR
00002453 86E0               <1>       XCHG  AH, AL             ; SWAP HIGH/LOW BYTES FOR WORD MOVE
00002455 F6C280             <1>       TEST  DL, 80H                  ; AGAIN, IS THIS XOR FUNCTION
00002458 7407               <1>       JZ    short _S10         ; NO, JUST STORE THE VALUES
0000245A 66338700200000     <1>       XOR   AX, [eDI+2000H]          ; FUNCTION WITH FIRST HALF LOW
                            <1> _S10:
00002461 66898700200000     <1>       MOV   [eDI+2000H], AX          ; STORE SECOND PORTION HIGH
00002468 6683C750           <1>       ADD   DI, 80             ; POINT TO NEXT LOCATION
0000246C FECE               <1>       DEC   DH
0000246E 75C4               <1>       JNZ   short S8           ; KEEP GOING
00002470 5E                 <1>       POP   eSI                ; RECOVER CODE POINTER
00002471 5F                 <1>       POP   eDI                ; RECOVER REGEN POINTER
00002472 47                 <1>       INC   eDI                ; POINT TO NEXT CHAR POSITION
00002473 47                 <1>       INC   eDI
00002474 E2BA               <1>       LOOP  S7                 ; MORE TO WRITE
00002476 C3                 <1>       retn
                            <1> 
                            <1> ; 04/07/2016
                            <1> ; 01/07/2016
                            <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
                            <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                            <1> ;----------------------------------------
                            <1> ; GRAPHICS READ
                            <1> ;----------------------------------------
                            <1> GRAPHICS_READ:
00002477 E8A3000000         <1>       CALL  S26                ; CONVERTED TO OFFSET IN REGEN
0000247C 89C6               <1>       MOV   eSI, eAX           ; SAVE IN SI
0000247E 81C600800B00       <1>       add   esi, 0B8000h       ; 01/07/2016
00002484 83EC08             <1>       SUB   eSP, 8             ; ALLOCATE SPACE FOR THE READ CODE POINT
00002487 89E5               <1>       MOV   eBP, eSP           ; POINTER TO SAVE AREA
                            <1> 
                            <1> ;-----      DETERMINE GRAPHICS MODES
00002489 B604               <1>       mov   dh, 4              ; number of passes ; 01/07/2016
0000248B 803D[C25E0000]06   <1>       CMP   byte [CRT_MODE], 6
00002492 7219               <1>       JC    short S12          ; MEDIUM RESOLUTION
                            <1> 
                            <1> ;-----      HIGH RESOLUTION READ
                            <1> ;-----      GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
                            <1>       ;MOV  DH,4               ; NUMBER OF PASSES
                            <1> S11:
00002494 8A06               <1>       MOV   AL, [eSI]          ; GET FIRST BYTE
00002496 884500             <1>       MOV   [eBP], AL          ; SAVE IN STORAGE AREA
00002499 45                 <1>       INC   eBP                ; NEXT LOCATION
0000249A 8A8600200000       <1>       MOV   AL, [eSI+2000H]          ; GET LOWER REGION BYTE
000024A0 884500             <1>       MOV   [eBP], AL          ; ADJUST AND STORE
000024A3 45                 <1>       INC   eBP
000024A4 83C650             <1>       ADD   eSI, 80                  ; POINTER INTO REGEN
000024A7 FECE               <1>       DEC   DH                 ; LOOP CONTROL
000024A9 75E9               <1>       JNZ   short S11          ; DO IT SOME MORE
000024AB EB1D               <1>       JMP   SHORT S14          ; GO MATCH THE SAVED CODE POINTS
                            <1> 
                            <1> ;-----      MEDIUM RESOLUTION READ
                            <1> S12:
000024AD 66D1E6             <1>       SAL   SI, 1              ; OFFSET*2 SINCE 2 BYTES/CHAR
                            <1>       ;MOV  DH, 4              ; NUMBER OF PASSES
                            <1> S13:
000024B0 E84D000000         <1>       CALL  S23                ; GET BYTES FROM REGEN INTO SINGLE SAVE
000024B5 81C6FE1F0000       <1>       ADD   eSI, 2000H-2       ; GO TO LOWER REGION
000024BB E842000000         <1>       CALL  S23                ; GET THIS PAIR INTO SAVE
000024C0 81EEB21F0000       <1>       SUB   eSI, 2000H-80+2          ; ADJUST POINTER BACK INTO UPPER
000024C6 FECE               <1>       DEC   DH
000024C8 75E6               <1>       JNZ   short S13          ; KEEP GOING UNTIL ALL 8 DONE
                            <1> 
                            <1> ;-----      SAVE AREA HAS CHARACTER IN IT, MATCH IT
                            <1> S14:                           ; FIND_CHAR
000024CA BF[B82C0100]       <1>       MOV   eDI, CRT_CHAR_GEN  ; ESTABLISH ADDRESSING
000024CF 83ED08             <1>       SUB   eBP, 8             ; ADJUST POINTER TO START OF SAVE AREA
000024D2 89EE               <1>       MOV   eSI, eBP
                            <1> S15:
000024D4 66B80001           <1>       mov   ax, 256                  ; NUMBER TO TEST AGAINST
                            <1> S16:
000024D8 56                 <1>       PUSH  eSI                ; SAVE SAVE AREA POINTER
000024D9 57                 <1>       PUSH  eDI                ; SAVE CODE POINTER
                            <1>       ;MOV  eCX, 4             ; NUMBER OF WORDS TO MATCH
                            <1>       ;REPE CMPSW              ; COMPARE THE 8 BYTES AS WORDS
000024DA A7                 <1>       cmpsd                    ; compare first 4 bytes
000024DB 7501               <1>       jne   short S17          ;
000024DD A7                 <1>       cmpsd                    ; compare last 4 bytes
                            <1> S17:
000024DE 5F                 <1>       POP   eDI                ; RECOVER THE POINTERS
000024DF 5E                 <1>       POP   eSI
                            <1>       ;JZ   short S18          ; IF ZERO FLAG SET, THEN MATCH OCCURRED
000024E0 7407               <1>       je    short S18
                            <1>       ;                        ; NO MATCH, MOVE ON TO NEXT
000024E2 83C708             <1>       ADD   eDI, 8             ; NEXT CODE POINT
000024E5 6648               <1>       dec   ax                 ; LOOP CONTROL
000024E7 75EF               <1>       JNZ   short S16          ; DO ALL OF THEM
                            <1> 
                            <1> ;-----      CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
                            <1> S18:
000024E9 83C408             <1>       ADD   eSP, 8             ; READJUST THE STACK, THROW AWAY SAVE
000024EC C3                 <1>       retn                     ; ALL DONE
                            <1> 
                            <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
                            <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                            <1> ;--------------------------------------------
                            <1> ; EXPAND BYTE
                            <1> ;  THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
                            <1> ;  OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
                            <1> ;  THE RESULT IS LEFT IN AX
                            <1> ;--------------------------------------------
                            <1> S21:
000024ED 6651               <1>       PUSH  CX                 ; SAVE REGISTER
```

```
                             <1>       ;MOV   CX, 8               ; SHIFT COUNT REGISTER FOR ONE BYTE
000024EF B108                <1>       mov    cl, 8
                             <1> S22:
000024F1 D0C8                <1>       ROR    AL,1                ; SHIFT BITS, LOW BIT INTO CARRY FLAG
000024F3 66D1DD              <1>       RCR    BP,1                ; MOVE CARRY FLAG (LOW BIT INTO RESULTS
000024F6 66D1FD              <1>       SAR    BP,1                ; SIGN EXTEND HIGH BIT (DOUBLE IT)
                             <1>       ;LOOP  S22                 ; REPEAT FOR ALL 8 BITS
000024F9 FEC9                <1>       dec    cl
000024FB 75F4                <1>       jnz    short S22
000024FD 6695                <1>       XCHG   AX, BP              ; MOVE RESULTS TO PARAMETER REGISTER
000024FF 6659                <1>       POP    CX                  ; RECOVER REGISTER
00002501 C3                  <1>       RETn                       ; ALL DONE
                             <1>
                             <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                             <1> ;--------------------------------------------------
                             <1> ; MED_READ_BYTE
                             <1> ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
                             <1> ;  COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
                             <1> ;  THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
                             <1> ;  POSITION IN THE SAVE AREA
                             <1> ; ENTRY --
                             <1> ;  SI,DS = POINTER TO REGEN AREA OF INTEREST
                             <1> ;  BX = EXPANDED FOREGROUND COLOR
                             <1> ;  BP = POINTER TO SAVE AREA
                             <1> ; EXIT --
                             <1> ;  SI AND BP ARE INCREMENTED
                             <1> ;----------------------------------------------------
                             <1> S23:
00002502 66AD                <1>       LODSW                      ; GET FIRST BYTE AND SECOND BYTES
00002504 86C4                <1>       XCHG   AL, AH              ; SWAP FOR COMPARE
00002506 66B900C0            <1>       MOV    CX, 0C000H          ; 2 BIT MASK TO TEST THE ENTRIES
0000250A B200                <1>       MOV    DL, 0               ; RESULT REGISTER
                             <1> S24:
0000250C 6685C8              <1>       TEST   AX, CX              ; IS THIS SECTION BACKCROUND?
0000250F 7401                <1>         JZ      short S25                ; IF ZERO, IT IS BACKGROUND (CARRY=0)
00002511 F9                  <1>       STC                        ; WASN'T, SO SET CARRY
                             <1> S25:
00002512 D0D2                <1>       RCL    DL, 1               ; MOVE THAT BIT INTO THE RESULT
00002514 66C1E902            <1>       SHR    CX, 2               ; MOVE THE MASK TO THE RIGHT BY 2 BITS
00002518 73F2                <1>       JNC    short S24           ; DO IT AGAIN IF MASK DIDN'T FALL OUT
0000251A 885500              <1>       MOV    [eBP], DL           ; STORE RESULT IN SAVE AREA
0000251D 45                  <1>       INC    eBP                 ; ADJUST POINTER
0000251E C3                  <1>       RETn                       ; ALL DONE
                             <1>
                             <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                             <1> ;-----------------------------------------
                             <1> ; V4_POSITION
                             <1> ;  THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
                             <1> ;  THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
                             <1> ;  INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
                             <1> ;  FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
                             <1> ;  BE DOUBLED.
                             <1> ; ENTRY -- NO REGISTERS,MEMORY LOCATION @CURSOR_POSN IS USED
                             <1> ; EXIT--
                             <1> ;  AX CONTAINS OFFSET INTO REGEN BUFFER
                             <1> ;-----------------------------------------
                             <1> S26:
0000251F 0FB705[56580100]    <1>       movzx  eax, word [CURSOR_POSN]    ; GET CURRENT CURSOR
                             <1> GRAPH_POSN:
00002526 53                  <1>       PUSH   eBX                 ; SAVE REGISTER
00002527 0FB6D8              <1>       movzx  ebx, al                    ; SAVE A COPY OF CURRENT CURSOR
0000252A A0[C45E0000]        <1>       MOV    AL, [CRT_COLS]             ; GET BYTES PER COLUMN
0000252F F6E4                <1>       MUL    AH                  ; MULTIPLY BY ROWS
00002531 66C1E002            <1>       SHL    AX, 2               ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
00002535 01D8                <1>       ADD    eAX, eBX            ; DETERMINE OFFSET
00002537 5B                  <1>       POP    eBX                 ; RECOVER POINTER
00002538 C3                  <1>       RETn                       ; ALL DONE
                             <1>
                             <1> ; 09/07/2016
                             <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                             <1> ;---------------------------------------------
                             <1> ; SET_COLOR
                             <1> ;     THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN COLOR,
                             <1> ;     AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION GRAPHICS
                             <1> ; INPUT
                             <1> ;     (BH) HAS COLOR ID
                             <1> ;           IF BH=0, THE BACKGROUND COLOR VALUE IS SET
                             <1> ;                  FROM THE LOW BITS OF BL (0-31)
                             <1> ;           IF BH=1, THE PALETTE SELECTION IS MADE
                             <1> ;                  BASED ON THE LOW BIT OF BL:
                             <1> ;                         0 = GREEN, RED, YELLOW FOR COLORS 1,2,3
                             <1> ;                         1 = BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
                             <1> ;     (BL) HAS THE COLOR VALUE TO BE USED
                             <1> ; OUTPUT
                             <1> ;     THE COLOR SELECTION IS UPDATED
                             <1> ;----------------------------------------------
                             <1> SET_COLOR:
00002539 803D[C25E0000]07    <1>         cmp     byte [CRT_MODE], 7      ; 09/07/2016
00002540 0F870EF0FFFF        <1>       ja     VIDEO_RETURN        ; nothing to do for VGA modes
                             <1>
                             <1>       ;MOV   DX, [ADDR_6845]            ; I/O PORT FOR PALETTE
                             <1>       ;mov   dx, 3D4h
                             <1>       ;ADD   DX,5                ; OVERSCAN PORT
00002546 66BAD903            <1>       mov    dx, 3D9h
0000254A A0[C55E0000]        <1>       MOV    AL, [CRT_PALETTE]   ; GET THE CURRENT PALETTE VALUE
0000254F 08FF                <1>       OR     BH, BH              ; IS THIS COLOR 0?
00002551 7512                <1>       JNZ    short M20           ; OUTPUT COLOR 1
                             <1>
                             <1> ;-----      HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
                             <1>
```

```
00002553 24E0                <1>       AND    AL, 0E0H            ; TURN OFF LOW 5 BITS OF CURRENT
00002555 80E31F              <1>       AND    BL, 01FH            ; TURN OFF HIGH 3 BITS OF INPUT VALUE
00002558 08D8                <1>       OR     AL, BL              ; PUT VALUE INTO REGISTER
                             <1> M19:                             ; OUTPUT THE PALETTE
0000255A EE                  <1>       OUT    DX, AL              ; OUTPUT COLOR SELECTION TO 3D9 PORT
0000255B A2[C55E0000]        <1>       MOV    [CRT_PALETTE], AL   ; SAVE THE COLOR VALUE
00002560 E9EFEFFFFF          <1>       JMP    VIDEO_RETURN
                             <1>
                             <1> ;-----       HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
                             <1>
                             <1> M20:
00002565 24DF                <1>       AND    AL, 0DFH            ; TURN OFF PALETTE SELECT BIT
00002567 D0EB                <1>       SHR    BL, 1               ; TEST THE LOW ORDER BIT OF BL
00002569 73EF                <1>       JNC    short M19           ; ALREADY DONE
0000256B 0C20                <1>       OR     AL, 20H                   ; TURN ON PALETTE SELECT BIT
0000256D EBEB                <1>       JMP    short M19           ; GO DO IT
                             <1>
                             <1> ; 09/07/2016
                             <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                             <1> ;-------------------------------------------
                             <1> ; READ DOT -- WRITE DOT
                             <1> ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
                             <1> ;  DOT AT THE INDICATED LOCATION
                             <1> ; ENTRY --
                             <1> ;   DX = ROW (0-199)      (THE ACTUAL VALUE DEPENDS ON THE MODE)
                             <1> ;   CX = COLUMN ( 0-639) ( THE VALUES ARE NOT RANGE CHECKED )
                             <1> ;   AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
                             <1> ;     REQUIRED FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
                             <1> ;     BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
                             <1> ;   DS = DATA SEGMENT
                             <1> ;   ES = REGEN SEGMENT
                             <1> ;
                             <1> ; EXIT
                             <1> ;     AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
                             <1> ;---------------------------------------------
                             <1>
                             <1> READ_DOT:
                             <1>       ; 09/07/2016
0000256F 8A25[C25E0000]      <1>       mov    ah,  [CRT_MODE]
00002575 80FC07              <1>       cmp    ah, 7 ; 6!?
00002578 760A                <1>       jna    short read_dot_cga
                             <1>
0000257A E8CB030000          <1>       call   vga_read_pixel
                             <1>       ; al = pixel value
0000257F E9D5EFFFFF          <1>       jmp    _video_return
                             <1>
                             <1> read_dot_cga:
                             <1>       ;je    VIDEO_RETURN ; 7
00002584 80FC04              <1>       cmp    ah, 4 ; graphics ?
00002587 0F82C7EFFFFF        <1>       jb     VIDEO_RETURN ; no, text mode, nothing to do
                             <1>
0000258D E855000000          <1>       CALL   R3                  ; DETERMINE BYTE POSITION OF DOT
00002592 8A06                <1>       MOV    AL, [eSI]           ; GET THE BYTE
00002594 20E0                <1>       AND    AL, AH              ; MASK OFF THE OTHER BITS IN THE BYTE
00002596 D2E0                <1>       SHL    AL, CL              ; LEFT JUSTIFY THE VALUE
00002598 88F1                <1>       MOV    CL, DH              ; GET NUMBER OF BITS IN RESULT
0000259A D2C0                <1>       ROL    AL, CL              ; RIGHT JUSTIFY THE RESULT
                             <1>       ;JMP   VIDEO_RETURN        ; RETURN FROM VIDEO I/O
0000259C 0FB6C0              <1>       movzx  eax, al
0000259F E9B5EFFFFF          <1>       jmp    _video_return
                             <1>
                             <1> ; 09/07/2016
                             <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                             <1>
                             <1> WRITE_DOT:
                             <1>       ; 09/07/2016
000025A4 8A25[C25E0000]      <1>       mov    ah, [CRT_MODE]
000025AA 80FC07              <1>       cmp    ah, 7 ; 6!?
000025AD 760A                <1>       jna    short write_dot_cga
                             <1>
000025AF E805030000          <1>       call   vga_write_pixel
000025B4 E99BEFFFFF          <1>       jmp    VIDEO_RETURN
                             <1>
                             <1> write_dot_cga:
                             <1>       ;je    VIDEO_RETURN ; 7
000025B9 80FC04              <1>       cmp    ah, 4 ; graphics ?
000025BC 0F8292EFFFFF        <1>       jb     VIDEO_RETURN ; no, text mode, nothing to do
                             <1>
                             <1>       ;PUSH  AX                  ; SAVE DOT VALUE
000025C2 6650                <1>       PUSH   AX                  ; TWICE
000025C4 E81E000000          <1>       CALL   R3                  ; DETERMINE BYTE POSITION OF THE DOT
000025C9 D2E8                <1>       SHR    AL, CL              ; SHIFT TO SET UP THE BITS FOR OUTPUT
000025CB 20E0                <1>       AND    AL, AH              ; STRIP OFF THE OTHER BITS
000025CD 8A0E                <1>       MOV    CL, [eSI]           ; GET THE CURRENT BYTE
000025CF 665B                <1>       POP    BX                  ; RECOVER XOR FLAG
000025D1 F6C380              <1>       TEST   BL, 80H                   ; IS IT ON
000025D4 750D                <1>       JNZ    short R2            ; YES, XOR THE DOT
000025D6 F6D4                <1>       NOT    AH                  ; SET MASK TO REMOVE THE INDICATED BITS
000025D8 20E1                <1>       AND    CL, AH
000025DA 08C8                <1>       OR     AL, CL              ; OR IN THE NEW VALUE OF THOSE BITS
                             <1> R1:                              ; FINISH_DOT
000025DC 8806                <1>       MOV    [eSI], AL           ; RESTORE THE BYTE IN MEMORY
                             <1>       ;POP   AX
000025DE E971EFFFFF          <1>       JMP    VIDEO_RETURN        ; RETURN FROM VIDEO I/O
                             <1> R2:                              ; XOR_DOT
000025E3 30C8                <1>       XOR    AL, CL              ; EXCLUSIVE OR THE DOTS
000025E5 EBF5                <1>       JMP    short R1            ; FINISH UP THE WRITING
                             <1>
                             <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
                             <1>
                             <1> ;----------------------------------------------
                             <1> ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
                             <1> ; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
                             <1> ; ENTRY --
                             <1> ;  DX = ROW VALUE (0-199)
                             <1> ;  CX = COLUMN VALUE (0-639)
                             <1> ; EXIT --
                             <1> ;  SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
                             <1> ;  AH = MASK TO STRIP OFF THE BITS OF INTEREST
                             <1> ;  CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
                             <1> ;  DH = # BITS IN RESULT
                             <1> ;  BX = MODIFIED
                             <1> ;-----------------------------------------------
                             <1> R3:
                             <1>
                             <1> ;-----       DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40
                             <1> ;-----        ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW )
                             <1>
000025E7 0FB7F0              <1>       movzx  esi, ax                   ; WILL SAVE AL AND AH DURING OPERATION
000025EA B028                <1>       MOV    AL, 40
000025EC F6E2                <1>       MUL    DL                 ; AX= ADDRESS OF START OF INDICATED ROW
000025EE A808                <1>       TEST   AL, 08H            ; TEST FOR EVEN/ODD ROW CALCULATED
000025F0 7404                <1>       JZ     short R4           ; JUMP IF EVEN ROW
000025F2 6605D81F            <1>       ADD    AX, 2000H-40       ; OFFSET TO LOCATION OF ODD ROWS ADJUST
                             <1> R4:                              ; EVEN_ROW
000025F6 6696                <1>       XCHG   SI, AX             ; MOVE POINTER TO (SI) AND RECOVER (AX)
000025F8 81C600800B00        <1>       add    esi, 0B8000h
000025FE 6689CA              <1>       MOV    DX, CX             ; COLUMN VALUE TO DX
                             <1>
                             <1> ;-----       DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
                             <1>
                             <1> ; SET UP THE REGISTERS ACCORDING TO THE MODE
                             <1> ; CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES )
                             <1> ; CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M )
                             <1> ; BL = MASK TO SELECT BITS FROM POINTED BYTE ( 80H/C0H FOR H/M )
                             <1> ; BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M )
                             <1>
00002601 66BBC002            <1>       MOV    BX, 2C0H
00002605 66B90203            <1>       MOV    CX, 302H           ; SET PARMS FOR MED RES
00002609 803D[C25E0000]06    <1>       CMP    byte [CRT_MODE], 6
00002610 7208                <1>       JC     short R5           ; HANDLE IF MED RES
00002612 66BB8001            <1>       MOV    BX, 180H
00002616 66B90307            <1>       MOV    CX, 703H           ; SET PARMS FOR HIGH RES
                             <1>
                             <1> ;-----       DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
                             <1> R5:
0000261A 20D5                <1>       AND    CH, DL             ; ADDRESS OF PEL WITHIN BYTE TO CH
                             <1>
                             <1> ;-----       DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
                             <1>
0000261C 66D3EA              <1>       SHR    DX, CL             ; SHIFT BY CORRECT AMOUNT
0000261F 6601D6              <1>       ADD    SI, DX             ; INCREMENT THE POINTER
00002622 88FE                <1>       MOV    DH, BH             ; GET THE # OF BITS IN RESULT TO DH
                             <1>
                             <1> ;-----       MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
                             <1>
00002624 28C9                <1>       SUB    CL, CL             ; ZERO INTO STORAGE LOCATION
                             <1> R6:
00002626 D0C8                <1>       ROR    AL, 1              ; LEFT JUSTIFY VALUE IN AL (FOR WRITE)
00002628 00E9                <1>       ADD    CL, CH             ; ADD IN THE BIT OFFSET VALUE
0000262A FECF                <1>       DEC    BH                 ; LOOP CONTROL
0000262C 75F8                <1>       JNZ    short R6           ; ON EXIT, CL HAS COUNT TO RESTORE BITS
0000262E 88DC                <1>       MOV    AH, BL             ;  GET MASK TO AH
00002630 D2EC                <1>       SHR    AH, CL             ;  MOVE THE MASK TO CORRECT LOCATION
00002632 C3                  <1>       RETn                       ;  RETURN WITH EVERYTHING SET UP
                             <1>
                             <1> load_dac_palette:
                             <1>       ; 29/07/2016
                             <1>       ; 23/07/2016
                             <1>       ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; (set_mode_vga)
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'load_dac_palette'
                             <1>       ;
                             <1>       ; Oracle VirtualBox 5.0.24 VGABios Source Code
                             <1>       ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
                             <1>       ;
                             <1>       ; INPUT -> AH = DAC selection number (3, 2 or 1)
                             <1>       ; OUTPUT -> ECX = 0, AX = 0
                             <1>       ; (Modifed registers: EAX, ECX, EDX, ESI)
                             <1>       ;
00002633 66BAC803            <1>       mov    dx, 3C8h  ; VGAREG_DAC_WRITE_ADDRESS
00002637 28C0                <1>       sub    al, al ; 0
00002639 EE                  <1>       out    dx, al ; 0 ; color index, always 0 at the beginning
0000263A 6642                <1>       inc    dx   ; 3C9h ; VGAREG_DAC_DATA
0000263C B900010000          <1>       mov    ecx, 256   ; always 256*3 values
                             <1>       ;push  esi
00002641 88E0                <1>       mov    al, ah
00002643 B43F                <1>       mov    ah, 3Fh      ; 3Fh except DAC selection number 3
00002645 3C02                <1>       cmp    al, 2
00002647 7414                <1>       je     short l_dac_p_2
00002649 7719                <1>       ja     short l_dac_p_3
0000264B 20C0                <1>       and    al, al
0000264D 7507                <1>       jnz    short l_dac_p_1
                             <1> l_dac_p_0:
0000264F BE[78270100]        <1>       mov    esi, palette0
00002654 EB15                <1>       jmp    short l_dac_p_4
                             <1> l_dac_p_1:
00002656 BE[38280100]        <1>       mov    esi, palette1
0000265B EB0E                <1>       jmp    short l_dac_p_4
                             <1> l_dac_p_2:
```

```
0000265D BE[F8280100]        <1>        mov    esi, palette2
00002662 EB07                <1>        jmp    short l_dac_p_4
                             <1> l_dac_p_3:
00002664 B4FF                <1>        mov    ah, 0FFh ; dac registers
00002666 BE[B8290100]        <1>        mov    esi, palette3
                             <1> l_dac_p_4:
0000266B AC                  <1>        lodsb
0000266C EE                  <1>        out    dx, al  ; Red
0000266D AC                  <1>        lodsb
0000266E EE                  <1>        out    dx, al ; Green
0000266F AC                  <1>        lodsb
00002670 EE                  <1>        out    dx, al ; Blue
00002671 20E4                <1>        and    ah, ah
00002673 7405                <1>        jz     short l_dac_p_5
00002675 FECC                <1>        dec    ah
00002677 E2F2                <1>        loop   l_dac_p_4
                             <1>        ;pop   esi
00002679 C3                  <1>        retn
                             <1> l_dac_p_5:
                             <1>        ; 29/07/2016
0000267A FEC9                <1>        dec    cl
0000267C 7407                <1>        jz     short l_dac_p_7
                             <1>        ;
0000267E 28C0                <1>        sub    al, al ; 0
                             <1> l_dac_p_6:
00002680 EE                  <1>        out    dx, al ; outb(VGAREG_DAC_DATA,0);
00002681 EE                  <1>        out    dx, al
00002682 EE                  <1>        out    dx, al
00002683 E2FB                <1>        loop   l_dac_p_6
                             <1> l_dac_p_7:
                             <1>        ;pop   esi
00002685 C3                  <1>        retn
                             <1>
                             <1> gray_scale_summing:
                             <1>        ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>        ; (set_mode_vga)
                             <1>        ; derived from 'Plex86/Bochs VGABios' source code
                             <1>        ; vgabios-0.7a (2011)
                             <1>        ; by the LGPL VGABios developers Team (2001-2008)
                             <1>        ; 'vgabios.c', 'biosfn_perform_gray_scale_summing'
                             <1>        ;
                             <1>        ; Oracle VirtualBox 5.0.24 VGABios Source Code
                             <1>        ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
                             <1>        ;
                             <1>
                             <1>        ; INPUT -> EBX = Start address (color index <= 255)
                             <1>        ;          ECX = Count (<= 256)
                             <1>        ; OUTPUT -> (E)CX = 0
                             <1>        ; (Modifed registers: EAX, ECX, EDX, EBX)
                             <1>
00002686 66BADA03            <1>        mov    dx, 3DAh ; VGAREG_ACTL_RESET
0000268A EC                  <1>        in     al, dx
0000268B 30C0                <1>        xor    al, al ; 0
0000268D 66BAC003            <1>        mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
00002691 EE                  <1>        out    dx, al ; clear bit 5
                             <1>                      ; (while loading palette registers)
                             <1>        ; set read address and switch to read mode
                             <1> g_s_s_1:
00002692 66BAC703            <1>        mov    dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
00002696 88D8                <1>        mov    al, bl
00002698 EE                  <1>        out    dx, al
                             <1>        ; get 6-bit wide RGB data values
                             <1>        ; intensity = (0.3*Red)+(0.59*Green)+(0.11*Blue)
                             <1>        ; i = ( ( 77*r + 151*g + 28*b ) + 0x80 ) >> 8;
00002699 66BAC903            <1>        mov    dx, 3C9h ; VGAREG_DAC_DATA
0000269D EC                  <1>        in     al, dx ; red
0000269E B44D                <1>        mov    ah, 77 ; 0.3* Red
000026A0 F6E4                <1>        mul    ah
000026A2 6650                <1>        push   ax
000026A4 EC                  <1>        in     al, dx ; green
000026A5 B497                <1>        mov    ah, 151  ; 0.59 * Green
000026A7 F6E4                <1>        mul    ah
000026A9 6650                <1>        push   ax
000026AB EC                  <1>        in     al, dx ; blue
000026AC B41C                <1>        mov    ah, 28 ; 0.11 * Blue
000026AE F6E4                <1>        mul    ah
000026B0 665A                <1>        pop    dx
000026B2 6601D0              <1>        add    ax, dx
000026B5 665A                <1>        pop    dx
000026B7 6601D0              <1>        add    ax, dx
000026BA 66058000            <1>        add    ax, 80h
000026BE B03F                <1>        mov    al, 3Fh
000026C0 38C4                <1>        cmp    ah, al
000026C2 7602                <1>        jna    short g_s_s_2
000026C4 88C4                <1>        mov    ah, al
                             <1> g_s_s_2:
000026C6 66BAC803            <1>        mov    dx, 3C8h  ; VGAREG_DAC_WRITE_ADDRESS
000026CA 88D8                <1>        mov    al, bl ; color index
000026CC EE                  <1>        out    dx, al
000026CD 88E0                <1>        mov    al, ah ; intensity
000026CF 6642                <1>        inc    dx ; 3C9h ; VGAREG_DAC_DATA
000026D1 EE                  <1>        out    dx, al ; R (R=G=B)
000026D2 88E0                <1>        mov    al, ah ; intensity
000026D4 EE                  <1>        out    dx, al ; G (R=G=B)
000026D5 88E0                <1>        mov    al, ah ; intensity
000026D7 EE                  <1>        out    dx, al ; B (R=G=B)
000026D8 6649                <1>        dec    cx
000026DA 7404                <1>        jz     short g_s_s_3
000026DC FEC3                <1>        inc    bl    ; next color index value
000026DE EBB2                <1>        jmp    short g_s_s_1
                             <1> g_s_s_3:
000026E0 66BADA03            <1>        mov    dx, 3DAh ; VGAREG_ACTL_RESET
000026E4 EC                  <1>        in     al, dx
```

```
000026E5 B020                <1>       mov    al, 20h
000026E7 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
000026EB EE                  <1>       out    dx, al ; 20h -> set bit 5
                             <1>                      ; (after loading palette regs)
000026EC C3                  <1>       retn
                             <1>
                             <1> vga_write_char_attr:
                             <1> vga_write_char_only:
                             <1>       ; 08/07/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'biosfn_write_char_attr'
                             <1>       ; 'biosfn_write_char_only'
                             <1>
                             <1>       ; INPUT ->
                             <1>       ; [CRT_MODE] = current video mode (>7)
                             <1>       ; CX = Count of characters to write
                             <1>       ; AL = Character to write
                             <1>       ; BL = Color of character
                             <1>       ; OUTPUT ->
                             <1>       ; Regen buffer updated
                             <1>
000026ED 8A25[C25E0000]      <1>       mov    ah, [CRT_MODE]
000026F3 668B15[56580100]    <1>       mov    dx, [CURSOR_POSN] ; cursor pos for page 0
                             <1>
000026FA BE[DE5E0000]        <1>       mov    esi, vga_modes
000026FF 89F7                <1>       mov    edi, esi
00002701 83C710              <1>       add    edi, vga_mode_count
                             <1> vga_wca_0:
00002704 AC                  <1>       lodsb
00002705 38E0                <1>       cmp    al, ah ; [CRT_MODE]
00002707 7405                <1>       je     short vga_wca_2
00002709 39FE                <1>       cmp    esi, edi
0000270B 72F7                <1>       jb     short vga_wca_0
                             <1> vga_wca_1:
0000270D C3                  <1>       retn   ; nothing to do
                             <1> vga_wca_2:
0000270E 83C64F              <1>       add    esi, vga_memmodel - (vga_modes + 1)
                             <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
                             <1>
                             <1>       ; biosfn_write_char_attr (car,page,attr,count)
                             <1>       ; AL = car, page = 0, BL = attr, CX = count
00002711 803E04              <1>       cmp    byte [esi], PLANAR4
00002714 741D                <1>       je     short vga_wca_planar
00002716 803E03              <1>       cmp    byte [esi], PLANAR1
00002719 7418                <1>       je     short vga_wca_planar
                             <1> vga_wca_linear8:
                             <1>       ; while((count-->0) && (xcurs<nbcols))
                             <1>       ; CX = count
0000271B 6621C9              <1>       and    cx, cx
0000271E 74ED                <1>       jz     short vga_wca_1
00002720 3A15[C45E0000]      <1>       cmp    dl, [CRT_COLS]
00002726 73E5                <1>       jnb    short vga_wca_1
                             <1>       ; write_gfx_char_lin(car,attr,xcurs,ycurs,nbcols);
                             <1>       ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
                             <1>       ; [CRT_COLS] = nbcols
00002728 E81E000000          <1>       call   write_gfx_char_lin
0000272D 6649                <1>       dec    cx ; count
0000272F FEC2                <1>       inc    dl ; xcurs
00002731 EBE8                <1>       jmp    short vga_wca_linear8
                             <1> vga_wca_planar:
                             <1>       ; while((count-->0) && (xcurs<nbcols))
                             <1>       ; CX = count
00002733 6621C9              <1>       and    cx, cx
00002736 74D5                <1>       jz     short vga_wca_1
00002738 3A15[C45E0000]      <1>       cmp    dl, [CRT_COLS]
0000273E 73CD                <1>       jnb    short vga_wca_1
                             <1>       ; write_gfx_char_pl4(car,attr,xcurs,ycurs,nbcols,cheight);
                             <1>       ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
                             <1>       ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight
00002740 E89D000000          <1>       call   write_gfx_char_pl4
00002745 6649                <1>       dec    cx ; count
00002747 FEC2                <1>       inc    dl ; xcurs
00002749 EBE8                <1>       jmp    short vga_wca_planar
                             <1>
                             <1> write_gfx_char_lin:
                             <1>       ; 08/08/2016
                             <1>       ; 31/07/2016
                             <1>       ; 08/07/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'write_gfx_char_lin'
                             <1>
                             <1>       ; write_gfx_char_lin(car,attr,xcurs,ycurs,nbcols)
                             <1>       ; INPUT ->
                             <1>       ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
                             <1>       ; [CRT_COLS] = nbcols
                             <1>       ; OUTPUT ->
                             <1>       ; Regen buffer updated
                             <1>
0000274B 51                  <1>       push   ecx
0000274C 53                  <1>       push   ebx
0000274D 52                  <1>       push   edx
0000274E 50                  <1>       push   eax
                             <1>       ; addr=xcurs*8+ycurs*nbcols*64;
                             <1>       ; 08/08/2016
0000274F 0FB6F0              <1>       movzx  esi, al ; car
00002752 0FB6C6              <1>       movzx  eax, dh ; ycurs
00002755 8A25[C45E0000]      <1>       mov    ah, [CRT_COLS] ; nbcols
```

```
0000275B F6E4                <1>       mul    ah
                             <1>       ;shl   ax, 6 ; * 64
0000275D 66C1E003            <1>       shl    ax, 3 ; * 8
                             <1>       ;sub   dh, dh
                             <1>       ;shl   dx, 3 ; xcurs * 8
                             <1>       ;movzx edi, dx
00002761 0FB6FA              <1>       movzx  edi, dl
00002764 66C1E703            <1>       shl    di, 3 ; xcurs * 8
00002768 30F6                <1>       xor    dh, dh
0000276A 8A15[C65E0000]      <1>       mov    dl, [CHAR_HEIGHT]
00002770 66F7E2              <1>       mul    dx
                             <1>       ; eax = ycurs*nbcols*8*[CHAR_HEIGHT]
00002773 01C7                <1>       add    edi, eax ; addr
00002775 81C700000A00        <1>       add    edi, 0A0000h
                             <1>       ;shl   si, 3 ; car * 8
0000277B 30E4                <1>       xor    ah, ah
0000277D A0[C65E0000]        <1>       mov    al, [CHAR_HEIGHT]
00002782 66F7E6              <1>       mul    si
00002785 6689C6              <1>       mov    si, ax
                             <1>       ;; esi = src = car * 8
                             <1>       ; esi = src = car * [CHAR_HEIGHT]
                             <1>       ; i = 0
                             <1>       ;add   esi, vgafont8 ; fdata [src+i]
                             <1>       ; 08/08/2016
00002788 A1[E6650100]        <1>       mov    eax, [VGA_INT43H]
0000278D 3D[B8420100]        <1>       cmp    eax, vgafont16
00002792 740F                <1>         je      short wgfxl_0
00002794 3D[B8340100]        <1>       cmp    eax, vgafont14
00002799 7408                <1>       je     short wgfxl_0
0000279B 81C6[B82C0100]      <1>       add    esi, vgafont8
000027A1 EB02                <1>       jmp    short wgfxl_1
                             <1> wgfxl_0:
000027A3 01C6                <1>       add    esi, eax
                             <1> wgfxl_1:
000027A5 28FF                <1>       sub    bh, bh ; i = 0
                             <1> wgfxl_2:
                             <1>       ; for(i=0;i<8;i++)
000027A7 57                  <1>       push   edi ; addr
000027A8 0FB605[C45E0000]    <1>       movzx  eax, byte [CRT_COLS] ; nbcols
000027AF F6E7                <1>       mul    bh ; nbcols*i
000027B1 66C1E003            <1>       shl    ax, 3 ; i*nbcols*8
                             <1>       ; dest=addr+i*nbcols*8;
000027B5 01C7                <1>       add    edi, eax ; dest + j ; j = 0
000027B7 B180                <1>       mov    cl, 80h ; mask = 0x80;
                             <1>       ; esi = fdata + src + i
                             <1>       ; for(j=0;j<8;j++)
000027B9 29D2                <1>       sub    edx, edx ; j = 0
                             <1> wgfxl_3:
000027BB 8A06                <1>       mov    al, [esi] ; al = fdata[src+i]
000027BD 20C8                <1>       and    al, cl ; if (fdata[src+i] & mask)
000027BF 7402                <1>       jz     short wgfxl_4  ; data = 0, zf = 1
000027C1 88D8                <1>       mov    al, bl ; data = attr;
                             <1> wgfxl_4:
                             <1>       ; write_byte(0xa000,dest+j,data);
000027C3 AA                  <1>       stosb  ; dest + j (+ 0A0000h)
                             <1>       ;inc   dl ; j++
                             <1>       ;cmp   dl, 8
000027C4 80FA07              <1>       cmp    dl, 7
000027C7 720E                <1>       jb     short wgfxl_5
000027C9 5F                  <1>       pop    edi
                             <1>       ; 08/08/2016
                             <1>       ;cmp   bh, 7
                             <1>       ;jnb   short wgfxl_6
000027CA FEC7                <1>       inc    bh ; i++
000027CC 3A3D[C65E0000]      <1>       cmp    bh, [CHAR_HEIGHT]
000027D2 7309                <1>       jnb    short wgfxl_6
000027D4 46                  <1>       inc    esi
000027D5 EBD0                <1>       jmp    short wgfxl_2
                             <1> wgfxl_5:
000027D7 D0E9                <1>       shr    cl, 1 ; mask >>= 1;
000027D9 FEC2                <1>       inc    dl ; j++
000027DB EBDE                <1>         jmp     short wgfxl_3
                             <1> wgfxl_6:
000027DD 58                  <1>       pop    eax
000027DE 5A                  <1>       pop    edx
000027DF 5B                  <1>       pop    ebx
000027E0 59                  <1>       pop    ecx
000027E1 C3                  <1>       retn
                             <1>
                             <1> write_gfx_char_pl4:
                             <1>       ; 08/08/2016
                             <1>       ; 08/07/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'write_gfx_char_pl4'
                             <1>
                             <1>       ; write_gfx_char_pl4(car,attr,xcurs,ycurs,nbcols,cheight)
                             <1>       ; INPUT ->
                             <1>       ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
                             <1>       ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight
                             <1>       ; OUTPUT ->
                             <1>       ; Regen buffer updated
                             <1>
000027E2 51                  <1>       push   ecx
000027E3 53                  <1>       push   ebx
000027E4 52                  <1>       push   edx
000027E5 50                  <1>       push   eax
                             <1> wgfxpl_f0:
                             <1>       ; switch(cheight)
000027E6 8A25[C65E0000]      <1>       mov    ah, [CHAR_HEIGHT]
000027EC 80FC10              <1>       cmp    ah, 16 ; case 16:
```

```
000027EF 7507                <1>       jne    short wgfxpl_f1
                             <1>       ; fdata = &vgafont16;
000027F1 BE[B8420100]        <1>       mov    esi, vgafont16
000027F6 EB13                <1>       jmp    short wgfxpl_f3
                             <1> wgfxpl_f1:
000027F8 80FC0E              <1>       cmp    ah, 14 ; case 14:
000027FB 7507                <1>       jne    short wgfxpl_f2
000027FD BE[B8340100]        <1>       mov    esi, vgafont14
00002802 EB07                <1>       jmp    short wgfxpl_f3
                             <1> wgfxpl_f2:
                             <1>       ; default:
                             <1>       ;  fdata = &vgafont8;
00002804 BE[B82C0100]        <1>       mov    esi, vgafont8
00002809 B408                <1>       mov    ah, 8
                             <1> wgfxpl_f3:
                             <1>       ; al = car
0000280B F6E4                <1>       mul    ah ; ah = cheight
0000280D 25FFFF0000          <1>       and    eax, 0FFFFh ; car * cheight
                             <1>       ; src = car * cheight;
00002812 01C6                <1>       add    esi, eax ; esi = fdata[src+i]
                             <1>       ; addr=xcurs*8+ycurs*nbcols*64;
00002814 88F0                <1>       mov    al, dh ; ycurs
00002816 8A25[C45E0000]      <1>       mov    ah, [CRT_COLS] ; nbcols
0000281C F6E4                <1>       mul    ah
                             <1>       ; 08/08/2016
                             <1>       ;shl   ax, 6 ; * 64
0000281E 66C1E003            <1>       shl    ax, 3 ; * 8
                             <1>       ;sub   dh, dh ; 0
                             <1>       ;shl   dx, 3 ; xcurs * 8
                             <1>       ;movzx edi, dx
00002822 0FB6FA              <1>       movzx  edi, dl
00002825 66C1E703            <1>       shl    di, 3 ; xcurs * 8
00002829 30F6                <1>       xor    dh, dh
0000282B 8A15[C65E0000]      <1>       mov    dl, [CHAR_HEIGHT]
00002831 66F7E2              <1>       mul    dx
                             <1>       ; eax = ycurs*nbcols*8*[CHAR_HEIGHT]
00002834 01C7                <1>       add    edi, eax ; addr
00002836 81C700000A00        <1>       add    edi, 0A0000h
                             <1>       ;
                             <1>       ; outw(VGAREG_SEQU_ADDRESS, 0x0f02);
                             <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0205);
0000283C 66BAC403            <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
00002840 66B8020F            <1>       mov    ax, 0F02h
00002844 66EF                <1>       out    dx, ax
00002846 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
0000284A 66B80502            <1>       mov    ax, 0205h
0000284E 66EF                <1>       out    dx, ax
                             <1>       ;
00002850 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002854 F6C380              <1>       test   bl, 80h ; if(attr&0x80)
00002857 7406                <1>       jz     short wgfxpl_f4 ; else
                             <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x1803);
00002859 66B80318            <1>       mov    ax, 1803h
0000285D EB04                <1>       jmp    short wgfxpl_f5
                             <1> wgfxpl_f4:
                             <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0003);
0000285F 66B80300            <1>       mov    ax, 0003h
                             <1> wgfxpl_f5:
00002863 66EF                <1>       out    dx, ax
                             <1>       ;
00002865 28FF                <1>       sub    bh, bh ; i = 0
                             <1> wgfxpl_0:
                             <1>       ; for(i=0;i<cheight;i++)
00002867 57                  <1>       push   edi ; addr
00002868 0FB605[C45E0000]    <1>       movzx  eax, byte [CRT_COLS] ; nbcols
0000286F F6E7                <1>       mul    bh ; nbcols*i
                             <1>       ; dest=addr+i*nbcols
00002871 01C7                <1>       add    edi, eax ; dest
00002873 B580                <1>       mov    ch, 80h ; mask = 0x80;
                             <1>       ; for(j=0;j<8;j++)
00002875 28C9                <1>       sub    cl, cl ; j = 0
                             <1> wgfxpl_1:
00002877 D2ED                <1>       shr    ch, cl ; mask=0x80>>j;
                             <1>       ;
                             <1>       ; outw(VGAREG_GRDC_ADDRESS, (mask << 8) | 0x08);
                             <1>       ; read_byte(0xa000,dest);
                             <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002879 88EC                <1>       mov    ah, ch
0000287B B008                <1>       mov    al, 8
0000287D 66EF                <1>       out    dx, ax
0000287F 8A07                <1>       mov    al, [edi] ; ? (io delay?)
                             <1>       ;
00002881 28C0                <1>       sub    al, al ; attr = 0
                             <1>       ; if (fdata[src+i] & mask)
00002883 842E                <1>       test   byte [esi], ch
00002885 7404                <1>       jz     short wgfxpl_2  ; zf = 1
                             <1>       ; write_byte(0xa000,dest,attr&0x0f);
00002887 88D8                <1>       mov    al, bl ; attr;
00002889 240F                <1>       and    al, 0Fh      ; attr&0x0f
                             <1> wgfxpl_2:
                             <1>       ; write_byte(0xa000,dest,0x00);
0000288B 8807                <1>       mov    [edi], al ; dest (+ 0A0000h)
0000288D FEC1                <1>       inc    cl ; j++
0000288F 80F908              <1>       cmp    cl, 8
00002892 72E3                <1>       jb     short wgfxpl_1
00002894 5F                  <1>       pop    edi
                             <1>       ; 08/08/2016
                             <1>       ;cmp   bh, 7
                             <1>       ;jnb   short wgfxpl_3
00002895 FEC7                <1>       inc    bh ; i++
00002897 3A3D[C65E0000]      <1>       cmp    bh, [CHAR_HEIGHT]
0000289D 7303                <1>       jnb    short wgfxpl_3
0000289F 46                  <1>       inc    esi
```

```
000028A0 EBC5                <1>       jmp    short wgfxpl_0
                             <1> wgfxpl_3:
                             <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
000028A2 66B808FF            <1>       mov    ax, 0FF08h
000028A6 66EF                <1>       out    dx, ax
000028A8 66B80500            <1>       mov    ax, 0005h
000028AC 66EF                <1>       out    dx, ax
000028AE 66B80300            <1>       mov    ax, 0003h
000028B2 66EF                <1>       out    dx, ax
                             <1>       ;
000028B4 58                  <1>       pop    eax
000028B5 5A                  <1>       pop    edx
000028B6 5B                  <1>       pop    ebx
000028B7 59                  <1>       pop    ecx
000028B8 C3                  <1>       retn
                             <1>
                             <1> vga_write_pixel:
                             <1>       ; 09/07/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'biosfn_write_pixel'
                             <1>
                             <1>       ; INPUT ->
                             <1>       ;      DX = row (0-239)
                             <1>       ;      CX = column (0-799)
                             <1>       ;      AL = pixel value
                             <1>       ;      (AH = [CRT_MODE])
                             <1>       ; OUTPUT ->
                             <1>       ;      none
                             <1>
000028B9 88C3                <1>       mov    bl, al ; pixel value
                             <1>       ;mov   ah, [CRT_MODE]
000028BB BE[DE5E0000]        <1>       mov    esi, vga_modes
000028C0 89F7                <1>       mov    edi, esi
000028C2 83C710              <1>       add    edi, vga_mode_count
                             <1> vga_wp_0:
000028C5 AC                  <1>       lodsb
000028C6 38E0                <1>       cmp    al, ah ; [CRT_MODE]
000028C8 7405                <1>       je     short vga_wp_1
000028CA 39FE                <1>       cmp    esi, edi
000028CC 72F7                <1>       jb     short vga_wp_0
000028CE C3                  <1>       retn   ; nothing to do
                             <1> vga_wp_1:
000028CF 83C64F              <1>       add    esi, vga_memmodel - (vga_modes + 1)
                             <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
000028D2 BF00000A00          <1>       mov    edi, 0A0000h
                             <1>       ;
000028D7 803E04              <1>       cmp    byte [esi], PLANAR4
000028DA 741D                <1>       je     short vga_wp_planar
000028DC 803E03              <1>       cmp    byte [esi], PLANAR1
000028DF 7418                <1>       je     short vga_wp_planar
                             <1> vga_wp_linear8:
                             <1>       ; addr=CX+DX*(read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS)*8);
000028E1 0FB605[C45E0000]    <1>       movzx  eax, byte [CRT_COLS] ; = [VGA_COLS] ; nbcols
000028E8 66C1E003            <1>       shl    ax, 3 ; * 8
000028EC 66F7E2              <1>       mul    dx
000028EF 50                  <1>       push   eax
                             <1>       ;mov   edi, 0A0000h
000028F0 6601CF              <1>       add    di, cx
000028F3 58                  <1>       pop    eax
000028F4 01C7                <1>       add    edi, eax ; addr
                             <1>       ; write_byte(0xa000,addr,AL);
000028F6 881F                <1>       mov    [edi], bl
000028F8 C3                  <1>       retn
                             <1> vga_wp_planar:
                             <1>       ; addr = CX/8+DX*read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS);
000028F9 0FB7C1              <1>       movzx  eax, cx
000028FC 66C1E803            <1>       shr    ax, 3 ; CX/8
00002900 50                  <1>       push   eax
00002901 28E4                <1>       sub    ah, ah ; 0
00002903 A0[C45E0000]        <1>       mov    al, [CRT_COLS] ; = [VGA_COLS] ; nbcols
00002908 66F7E2              <1>       mul    dx
                             <1>       ;mov   edi, 0A0000h
0000290B 6601C7              <1>         add     di, ax
0000290E 58                  <1>       pop    eax
0000290F 01C7                <1>       add    edi, eax ; addr
00002911 80E107              <1>       and    cl, 7
00002914 B580                <1>       mov    ch, 80h ; mask
00002916 D2ED                <1>       shr    ch, cl       ; mask = 0x80 >> (CX & 0x07);
                             <1>
                             <1>       ; outw(VGAREG_GRDC_ADDRESS, (mask << 8) | 0x08);
00002918 66BACE03            <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
0000291C 88EC                <1>       mov    ah, ch
0000291E B008                <1>       mov    al, 8
00002920 66EF                <1>       out    dx, ax
                             <1>       ; outw(VGAREG_GRDC_ADDRESS, 0x0205);
00002922 66B80502            <1>       mov    ax, 0205h
00002926 66EF                <1>       out    dx, ax
                             <1>       ; data = read_byte(0xa000,addr);
00002928 8A07                <1>       mov    al, [edi] ; (delay?)
                             <1>       ; if (AL & 0x80)
                             <1>       ; {
                             <1>       ;  outw(VGAREG_GRDC_ADDRESS, 0x1803);
                             <1>       ; }
0000292A F6C380              <1>       test   bl, 80h
0000292D 7406                <1>       jz     short vga_wp_2
0000292F 66B80318            <1>       mov    ax, 1803h
00002933 66EF                <1>       out    dx, ax
                             <1> vga_wp_2:
                             <1>       ; write_byte(0xa000,addr,AL);
00002935 881F                <1>       mov    [edi], bl
```

```
                             <1>       ;
                             <1>       ;mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002937 66B808FF            <1>       mov   ax, 0FF08h
0000293B 66EF                <1>       out   dx, ax
0000293D 66B80500            <1>       mov   ax, 0005h
00002941 66EF                <1>       out   dx, ax
00002943 66B80300            <1>       mov   ax, 0003h
00002947 66EF                <1>       out   dx, ax
                             <1>       ;
00002949 C3                  <1>       retn
                             <1>
                             <1> vga_read_pixel:
                             <1>       ; 09/07/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'biosfn_read_pixel'
                             <1>
                             <1>       ; INPUT ->
                             <1>       ;     DX = row (0-239)
                             <1>       ;     CX = column (0-799)
                             <1>       ;     (AH = [CRT_MODE])
                             <1>       ; OUTPUT ->
                             <1>       ;     AL = pixel value
                             <1>
                             <1>       ;mov   ah, [CRT_MODE]
0000294A BE[DE5E0000]        <1>       mov   esi, vga_modes
0000294F 89F7                <1>       mov   edi, esi
00002951 83C710              <1>       add   edi, vga_mode_count
                             <1> vga_rp_0:
00002954 AC                  <1>       lodsb
00002955 38E0                <1>       cmp   al, ah ; [CRT_MODE]
00002957 7405                <1>       je    short vga_rp_1
00002959 39FE                <1>       cmp   esi, edi
0000295B 72F7                <1>       jb    short vga_rp_0
0000295D C3                  <1>       retn  ; nothing to do
                             <1> vga_rp_1:
0000295E 83C64F              <1>       add   esi, vga_memmodel - (vga_modes + 1)
                             <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
00002961 BF00000A00          <1>       mov   edi, 0A0000h
                             <1>       ;
00002966 803E04              <1>       cmp   byte [esi], PLANAR4
00002969 741D                <1>       je    short vga_rp_planar
0000296B 803E03              <1>       cmp   byte [esi], PLANAR1
0000296E 7418                <1>       je    short vga_rp_planar
                             <1> vga_rp_linear8:
                             <1>       ; addr=CX+DX*(read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS)*8);
00002970 0FB605[C45E0000]    <1>       movzx eax, byte [CRT_COLS] ; = [VGA_COLS] ; nbcols
00002977 66C1E003            <1>       shl   ax, 3 ; * 8
0000297B 66F7E2              <1>       mul   dx
0000297E 50                  <1>       push  eax
                             <1>       ;mov   edi, 0A0000h
0000297F 6601CF              <1>       add   di, cx
00002982 58                  <1>       pop   eax
00002983 01C7                <1>       add   edi, eax ; addr
                             <1>       ; attr=read_byte(0xa000,addr);
00002985 8A07                <1>       mov   al, [edi] ; pixel value
00002987 C3                  <1>       retn
                             <1> vga_rp_planar:
                             <1>       ; addr = CX/8+DX*read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS);
00002988 0FB7C1              <1>       movzx eax, cx
0000298B 66C1E803            <1>       shr   ax, 3 ; CX/8
0000298F 50                  <1>       push  eax
00002990 28E4                <1>       sub   ah, ah ; 0
00002992 A0[C45E0000]        <1>       mov   al, [CRT_COLS] ; = [VGA_COLS] ; nbcols
00002997 66F7E2              <1>       mul   dx
                             <1>       ;mov   edi, 0A0000h
0000299A 6601C7              <1>         add     di, ax
0000299D 58                  <1>       pop   eax
0000299E 01C7                <1>       add   edi, eax ; addr
000029A0 80E107              <1>       and   cl, 7
000029A3 B580                <1>       mov   ch, 80h ; mask
000029A5 D2ED                <1>       shr   ch, cl       ; mask = 0x80 >> (CX & 0x07);
                             <1>       ; attr = 0x00;
000029A7 30DB                <1>       xor   bl, bl ; attr = bl = 0,
000029A9 30C9                <1>       xor   cl, cl ; i = cl = 0
                             <1>       ; for(i=0;i<4;i++)
                             <1>             ; {
                             <1>             ;  outw(VGAREG_GRDC_ADDRESS, (i << 8) | 0x04);
                             <1>             ;  data = read_byte(0xa000,addr) & mask;
                             <1>             ;  if (data > 0) attr |= (0x01 << i);
                             <1>             ; }
                             <1> vga_rp_2:
000029AB 88CC                <1>       mov   ah, cl ; i << 8
000029AD B004                <1>       mov   al, 4  ; | 0x04
000029AF 66BACE03            <1>       mov   dx, 3CEh ; VGAREG_GRDC_ADDRESS
000029B3 66EF                <1>       out   dx, ax
                             <1>       ; data = read_byte(0xa000,addr) & mask;
000029B5 8A07                <1>       mov   al, [edi]
000029B7 20E8                <1>       and   al, ch ; & mask
                             <1>       ; if (data > 0) attr |= (0x01 << i);
000029B9 08C0                <1>       or    al, al
000029BB 7408                <1>       jz    short vga_rp_3 ; al = 0
000029BD B701                <1>       mov   bh, 1
000029BF D2E7                <1>       shl   bh, cl ; (0x01 << i)
000029C1 08FB                <1>       or    bl, bh ; attr |= (0x01 << i)
000029C3 88D8                <1>       mov   al, bl ; pixel value
                             <1> vga_rp_3:
000029C5 C3                  <1>       retn
                             <1>
                             <1> vga_beeper:
                             <1>       ; 04/08/2016  (TRDOS 386 = TRDOS v2.0)
```

```
000029C6 FB                  <1>       sti
                             <1>       ;mov   bh, [ACTIVE_PAGE]
000029C7 E9CFF3FFFF          <1>         jmp     beeper_gfx
                             <1>
                             <1> vga_write_teletype:
                             <1>       ; 09/12/2017
                             <1>       ; 06/08/2016
                             <1>       ; 04/08/2016
                             <1>       ; 01/08/2016
                             <1>       ; 31/07/2016
                             <1>       ; 09/07/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'biosfn_write_teletype'
                             <1>       ; 'biosfn_write_char_only'
                             <1>
                             <1>       ; INPUT ->
                             <1>       ; [CRT_MODE] = current video mode (>7)
                             <1>       ; AL = Character to write
                             <1>       ; BL = Color of character
                             <1>       ; OUTPUT ->
                             <1>       ; Regen buffer updated
                             <1>
                             <1>       ; biosfn_write_teletype (car, page, attr, flag)
                             <1>       ; car = character (AL)
                             <1>       ; page = 0
                             <1>       ; attr = color (BL)
                             <1>       ; 'flag' not used
                             <1>
000029CC 8A25[C25E0000]      <1>       mov    ah, [CRT_MODE]
000029D2 88C7                <1>       mov    bh, al ; character
000029D4 668B15[56580100]    <1>       mov    dx, [CURSOR_POSN] ; cursor pos for page 0
                             <1>
000029DB BE[E65E0000]        <1>       mov    esi, vga_g_modes
000029E0 89F7                <1>       mov    edi, esi
000029E2 83C708              <1>       add    edi, vga_g_mode_count
                             <1> vga_wtty_0:
000029E5 AC                  <1>       lodsb
000029E6 38E0                <1>       cmp    al, ah ; [CRT_MODE]
000029E8 7405                <1>       je     short vga_wtty_2
000029EA 39FE                <1>       cmp    esi, edi
000029EC 72F7                <1>       jb     short vga_wtty_0
                             <1> vga_wtty_1:
000029EE C3                  <1>       retn   ; nothing to do
                             <1> vga_wtty_2:
000029EF 80FF07              <1>       cmp    bh, 07h ; bell (beep)
000029F2 74D2                <1>       je     short vga_beeper  ; u11
000029F4 80FF08              <1>       cmp    bh, 08h ; backspace
000029F7 7508                <1>       jne    short vga_wtty_3
                             <1>       ; if(xcurs>0)xcurs--;
000029F9 08D2                <1>       or     dl, dl ; xcurs (column)
000029FB 74F1                <1>       jz     short vga_wtty_1
000029FD FECA                <1>       dec    dl ; xcurs--;
000029FF EB59                <1>         jmp     short vga_wtty_12
                             <1> vga_wtty_3:
00002A01 80FF0D              <1>       cmp    bh, 0Dh ; carriage return (\r)
00002A04 7504                <1>       jne    short vga_wtty_4
                             <1>       ; xcurs=0;
00002A06 28D2                <1>       sub    dl, dl ; 0
00002A08 EB50                <1>         jmp     short vga_wtty_12
                             <1> vga_wtty_4:
00002A0A 80FF0A              <1>       cmp    bh, 0Ah ; new line (\n)
00002A0D 7504                <1>       jne    short vga_wtty_5
                             <1>       ; ycurs++;
00002A0F FEC6                <1>       inc    dh ; next row
00002A11 EB62                <1>         jmp     short vga_wtty_11
                             <1> vga_wtty_5:
00002A13 80FF09              <1>       cmp    bh, 09h ; tab stop
00002A16 7527                <1>       jne    short vga_wtty_8
00002A18 88D0                <1>       mov    al, dl
                             <1>       ;cbw
00002A1A 30E4                <1>       xor    ah, ah ; 09/12/2017
00002A1C B108                <1>       mov    cl, 8
00002A1E F6F1                <1>       div    cl
00002A20 28E1                <1>       sub    cl, ah
                             <1>       ;
00002A22 B720                <1>       mov    bh, 20h ; space
                             <1> vga_wtty_6: ; tab stop loop
00002A24 6651                <1>       push   cx
00002A26 6653                <1>       push   bx
00002A28 E812000000          <1>       call   vga_wtty_8
00002A2D 665B                <1>       pop    bx  ; bh = character, bl = color
00002A2F 6659                <1>       pop    cx
00002A31 FEC9                <1>       dec    cl
00002A33 7409                <1>       jz     short vga_wtty_7
00002A35 668B15[56580100]    <1>       mov    dx, [CURSOR_POSN] ; new cursor position (pg 0)
00002A3C EBE6                <1>       jmp    short vga_wtty_6
                             <1> vga_wtty_7:
00002A3E C3                  <1>       retn
                             <1>       ;
                             <1> vga_wtty_8:
00002A3F 83C64F              <1>       add    esi, vga_g_memmodel - (vga_g_modes + 1)
                             <1>       ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
00002A42 BF00000A00          <1>       mov    edi, 0A0000h
                             <1>       ;
00002A47 88F8                <1>       mov    al, bh ; character
                             <1>       ;
00002A49 803E04              <1>       cmp    byte [esi], PLANAR4
00002A4C 7414                <1>       je     short vga_wtty_planar
00002A4E 803E03              <1>       cmp    byte [esi], PLANAR1
00002A51 740F                <1>       je     short vga_wtty_planar
```

```
                             <1> vga_wtty_linear8:
                             <1>       ; write_gfx_char_lin(car,attr,xcurs,ycurs,nbcols);
                             <1>       ; AL = car, BL = attr (color), DL = xcurs, DH = ycurs,
                             <1>       ; [CRT_COLS] = nbcols
00002A53 E8F3FCFFFF          <1>       call   write_gfx_char_lin
00002A58 EB0D                <1>       jmp    short vga_wtty_9
                             <1>
                             <1> vga_wtty_12:
                             <1>       ; 09/07/2016
                             <1>       ; set cursor position
                             <1>       ; NOTE: Hardware cursor position will not be set
                             <1>       ;   in any VGA modes (>7)
                             <1>       ;   But, cursor position will be saved into
                             <1>       ;   [CURSOR_POSN].
                             <1>       ;   TRDOS 386 (TRDOS v2.0) uses only one page
                             <1>       ;   (page 0) for all graphics modes.
                             <1>
00002A5A 668915[56580100]    <1>       mov    [CURSOR_POSN], dx ; save cursor pos for pg 0
                             <1>       ; 04/08/2016
                             <1>       ;mov   bh, [ACTIVE_PAGE] ; = 0
                             <1>       ;call  _set_cpos
00002A61 C3                  <1>       retn
                             <1>
                             <1> vga_wtty_planar:
                             <1>       ; write_gfx_char_pl4(car,attr,xcurs,ycurs,nbcols,cheight);
                             <1>       ; AL = car, BL = attr (color), DL = xcurs, DH = ycurs,
                             <1>       ; [CRT_COLS]= nbcols, [CHAR_HEIGHT] = cheight
00002A62 E87BFDFFFF          <1>       call   write_gfx_char_pl4
                             <1> vga_wtty_9:
00002A67 FEC2                <1>       inc    dl ; xcurs++;
                             <1> vga_wtty_10:
                             <1>       ; Do we need to wrap ?
                             <1>       ; if(xcurs==nbcols)
00002A69 3A15[C45E0000]      <1>       cmp    dl, [CRT_COLS] ; [VGA_COLS]
00002A6F 7204                <1>       jb     short vga_wtty_11 ; no
00002A71 28D2                <1>       sub    dl, dl ; xcurs=0;
00002A73 FEC6                <1>       inc    dh ;  ycurs++;
                             <1> vga_wtty_11:
                             <1>       ; Do we need to scroll ?
                             <1>       ; if(ycurs==nbrows)
00002A75 3A35[CA5E0000]      <1>       cmp    dh, [VGA_ROWS]
00002A7B 72DD                <1>       jb     short vga_wtty_12 ; no
                             <1>       ;
                             <1>       ; biosfn_scroll (nblines,attr,rul,cul,rlr,clr,page,dir)
                             <1>       ; al = nblines = 1, bl = attr (color) = 0
                             <1>       ; ch = rul, cl = cul, dh = rlr, dl = clr, page = 0
                             <1>       ; dir = SCROLL_UP
                             <1>
00002A7D B001                <1>       mov    al, 1
00002A7F 28DB                <1>       sub    bl, bl ; 0 ; blank/black line (attr=0) will be used
00002A81 6629C9              <1>       sub    cx, cx ; 0,0
                             <1>
                             <1>       ; 06/08/2016
00002A84 8A35[CA5E0000]      <1>       mov    dh, [VGA_ROWS]
00002A8A FECE                <1>       dec    dh ; nbrows -1
                             <1>
00002A8C 6652                <1>       push   dx     ; 04/08/2016
00002A8E 8A15[C45E0000]      <1>       mov    dl, [CRT_COLS]
00002A94 FECA                <1>       dec    dl ; nbcols -1
                             <1>
00002A96 8A25[C25E0000]      <1>       mov    ah, [CRT_MODE]
                             <1>
                             <1>       ; biosfn_scroll(0x01,0x00,0,0,nbrows-1,nbcols-1,page,SCROLL_UP);
00002A9C E808F5FFFF          <1>       call   vga_graphics_up
                             <1>       ; 04/08/2016
00002AA1 665A                <1>       pop    dx
                             <1>       ;dec   dh ; ycurs-=1
00002AA3 EBB5                <1>       jmp    short vga_wtty_12
                             <1>
                             <1> font_setup:
                             <1>       ; 09/07/2016
                             <1>       ; character generator (font loading) functions
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'int10_func'
                             <1>
                             <1>       ; AX = 1100H ; Load User-Defined Font (EGA/VGA)
                             <1>       ;
                             <1>         ; BH    height of each character (bytes per character definition)
                             <1>         ; (BL   font block to load (EGA: 0-3; VGA: 0-7))
                             <1>       ; CX    number of characters to redefine (<=256)
                             <1>         ; DX    ASCII code of the first character defined at ES:BP
                             <1>         ; EBP      address of font-definition information
                             <1>       ;      (in user's memory space)
                             <1>
                             <1>       ; case 0x11:
                             <1>       ; switch(GET_AL())
                             <1>              ; {
                             <1>       ; case 0x00:
                             <1>         ; case 0x10:
                             <1>         ; biosfn_load_text_user_pat(GET_AL(),ES,BP,CX,DX,GET_BL(),GET_BH());
                             <1>         ; break;
                             <1>
                             <1>       ; AX = 1110H ; Load and Activate User-Defined Font (EGA/VGA)
00002AA5 08C0                <1>       or     al, al ; 0
00002AA7 7404                <1>       jz     short font_setup_0
00002AA9 3C10                <1>       cmp    al, 10h
00002AAB 7511                <1>       jne    short font_setup_1
                             <1> font_setup_0:
00002AAD E8B7000000          <1>       call   transfer_user_fonts
00002AB2 721C                <1>       jc     short font_setup_error
```

```
00002AB4 E8C2000000          <1>       call   load_text_user_pat
00002AB9 E996EAFFFF          <1>         jmp     VIDEO_RETURN
                             <1> font_setup_1:
                             <1>       ; AX = 1101H ; Load ROM 8x14 Character Set (EGA/VGA)
                             <1>       ; case 0x01:
                             <1>         ; case 0x11:
                             <1>         ; biosfn_load_text_8_14_pat(GET_AL(),GET_BL());
                             <1>         ; break;
00002ABE 3C01                <1>       cmp    al, 1
00002AC0 7404                <1>       je     short font_setup_2
00002AC2 3C11                <1>       cmp    al, 11h
00002AC4 7511                <1>       jne    short font_setup_3
                             <1> font_setup_2:
                             <1>       ; AX = 1111H ; Load and Activate ROM 8x14 Character Set (EGA/VGA)
                             <1>       ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
00002AC6 E8EE010000          <1>       call   load_text_8_14_pat
00002ACB E984EAFFFF          <1>         jmp     VIDEO_RETURN
                             <1> font_setup_error:
00002AD0 29C0                <1>       sub    eax, eax ; 0 -> fonts could not be loaded
00002AD2 E982EAFFFF          <1>       jmp    _video_return
                             <1> font_setup_3:
                             <1>       ; AX = 1102H ; Load ROM 8x8 Character Set (EGA/VGA)
                             <1>       ; case 0x02:
                             <1>         ; case 0x12:
                             <1>         ; biosfn_load_text_8_8_pat(GET_AL(),GET_BL());
                             <1>         ; break;
00002AD7 3C02                <1>       cmp    al, 2
00002AD9 7404                <1>       je     short font_setup_4
00002ADB 3C12                <1>       cmp    al, 12h
00002ADD 750A                <1>       jne    short font_setup_5
                             <1> font_setup_4:
                             <1>       ; AX = 1112H ; Load and Activate ROM 8x8 Character Set (EGA/VGA)
                             <1>       ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
00002ADF E805020000          <1>       call   load_text_8_8_pat
00002AE4 E96BEAFFFF          <1>         jmp     VIDEO_RETURN
                             <1> font_setup_5:
                             <1>       ; AX = 1104H ; Load ROM 8x16 Character Set (EGA/VGA)
                             <1>       ; case 0x04:
                             <1>         ; case 0x14:
                             <1>         ; biosfn_load_text_8_16_pat(GET_AL(),GET_BL());
                             <1>         ; break;
00002AE9 3C04                <1>       cmp    al, 4
00002AEB 7404                <1>       je     short font_setup_6
00002AED 3C14                <1>       cmp    al, 14h
00002AEF 750A                <1>       jne    short font_setup_7
                             <1> font_setup_6:
                             <1>       ; AX = 1114H ; Load and Activate ROM 8x16 Character Set (EGA/VGA)
                             <1>       ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
00002AF1 E823020000          <1>       call   load_text_8_16_pat
00002AF6 E959EAFFFF          <1>         jmp     VIDEO_RETURN
                             <1> font_setup_7:
                             <1>       ; Note: AX=1120h (Setup INT 1Fh, EXT_PTR) is not needed
                             <1>       ; for TRDOS 386 (TRDIOS v2.0) video functionality;
                             <1>       ; because, originally EXT_PTR (font address) was used for
                             <1>       ; chars 80h to 0FFh (after the first 128 ASCII char fonts), for
                             <1>       ; CGA graphics mode; currenty, 'vgafont8' address has 256 chars!
                             <1>       ;
                             <1>       ; case 0x20:
                             <1>         ; biosfn_load_gfx_8_8_chars(ES,BP);
                             <1>         ; break;
                             <1>       ; case 0x21:
                             <1>         ; biosfn_load_gfx_user_chars(ES,BP,CX,GET_BL(),GET_DL());
                             <1>         ; break;
                             <1>       ; AX = 1121H ; Setup User-Defined Font for Graphics Mode (VGA)
                             <1>       ; BL   screen rows code: 00H = user-specified (in DL)
                             <1>         ;                        01H = 14 rows
                             <1>         ;                        02H = 25 rows
                             <1>         ;                        03H = 43 rows
                             <1>         ; CX   bytes per character definition
                             <1>         ; DL   (when BL=0) custom number of character rows on screen
                             <1>         ; EBP  address of font-definition information (user's mem space)
                             <1>
00002AFB 3C21                <1>       cmp    al, 21h
00002AFD 751A                <1>       jne    short font_setup_9
                             <1>
                             <1>       ; TRDOS 386 modification !
                             <1>       ; dh = 0 -> 256 characters
                             <1>       ; dh = 80h -> 128 characters
                             <1>       ; (If DH <> 0 and DH <> 80h -> invalid)
00002AFF 20F6                <1>       and    dh, dh
00002B01 7405                <1>       jz     short font_setup_8 ; 256 characters
00002B03 80FE80              <1>       cmp    dh, 80h ; 128 characters
00002B06 75C8                <1>       jne    short font_setup_error ; invalid !
                             <1> font_setup_8:
00002B08 E85C000000          <1>       call   transfer_user_fonts
00002B0D 72C1                <1>       jc     short font_setup_error
                             <1>       ; ebp = user's font data address in system's memory space
00002B0F E836020000          <1>       call   load_gfx_user_chars
00002B14 E93BEAFFFF          <1>         jmp     VIDEO_RETURN
                             <1> font_setup_9:
                             <1>       ; case 0x22:
                             <1>         ; biosfn_load_gfx_8_14_chars(GET_BL());
                             <1>         ; break;
00002B19 3C22                <1>       cmp    al, 22h
00002B1B 750A                <1>       jne    short font_setup_10
00002B1D E866020000          <1>       call   load_gfx_8_14_chars
00002B22 E92DEAFFFF          <1>         jmp     VIDEO_RETURN
                             <1> font_setup_10:
                             <1>       ; case 0x23:
                             <1>         ; biosfn_load_gfx_8_8_dd_chars(GET_BL());
                             <1>         ; break;
00002B27 3C23                <1>       cmp    al, 23h
00002B29 750A                <1>       jne    short font_setup_11
```

```
00002B2B E899020000           <1>       call   load_gfx_8_8_chars
00002B30 E91FEAFFFF           <1>         jmp     VIDEO_RETURN
                              <1> font_setup_11:
                              <1>       ; case 0x24:
                              <1>         ; biosfn_load_gfx_8_16_chars(GET_BL());
                              <1>         ; break;
00002B35 3C24                 <1>       cmp    al, 24h
00002B37 750A                 <1>       jne    short font_setup_12
00002B39 E8CC020000           <1>       call   load_gfx_8_16_chars
00002B3E E911EAFFFF           <1>         jmp     VIDEO_RETURN
                              <1> font_setup_12:
                              <1>       ; case 0x30:
                              <1>         ; biosfn_get_font_info(GET_BH(),&ES,&BP,&CX,&DX);
                              <1>         ; break;
00002B43 3C30                 <1>       cmp    al, 30h
00002B45 750A                 <1>       jne    short font_setup_13
00002B47 E8FF020000           <1>       call   get_font_info
                              <1>       ; eax = return value (info: 4 bytes for 4 parms)
                              <1>       ; eax = 0 -> invalid function (input)
00002B4C E908EAFFFF           <1>         jmp     _video_return
                              <1> font_setup_13:
00002B51 3C03                 <1>       cmp    al, 03h ; AX = 1103h
00002B53 750D                 <1>       jne    short font_setup_14
                              <1>       ; biosfn_set_text_block_specifier:
                              <1>       ; BL = font block selector code
                              <1>       ; NOTE: TRDOS 386 only uses and sets font block 0
                              <1>       ; (It is as BL = 0 for TRDOS 386)
00002B55 66BAC403             <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
                              <1>       ;mov   ah, bl
00002B59 28E4                 <1>       sub    ah, ah ; 0
                              <1>       ;mov   al, 03h
00002B5B 66EF                 <1>       out    dx, ax
00002B5D E9F2E9FFFF           <1>       jmp     VIDEO_RETURN
                              <1>
                              <1> font_setup_14:
00002B62 29C0                 <1>       sub    eax, eax ; 0 = invalid function
00002B64 E9F0E9FFFF           <1>         jmp     _video_return
                              <1>
                              <1> transfer_user_fonts:
                              <1>       ; 09/07/2016
                              <1>       ;and   ecx, 0FFFFh
                              <1>       ; ECX = byte count
                              <1>       ;push  ecx
00002B69 89EE                 <1>       mov    esi, ebp ; user buffer
00002B6B BF00000700           <1>       mov    edi, Cluster_Buffer  ; system buffer
00002B70 E84EBC0000           <1>       call   transfer_from_user_buffer
                              <1>       ;pop   ecx
                              <1>       ; ecx = transfer (byte) count = character count
00002B75 BD00000700           <1>       mov    ebp, Cluster_Buffer
                              <1>       ; jc   VIDEO_RETURN -> failed
00002B7A C3                   <1>       retn
                              <1>
                              <1> load_text_user_pat:
                              <1>       ; 26/07/2016
                              <1>       ; 09/07/2016
                              <1>       ; load user defined (EGA/VGA) text fonts
                              <1>       ;
                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
                              <1>       ; vgabios-0.7a (2011)
                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
                              <1>       ; 'vgabios.c', 'biosfn_load_text_user_pat'
                              <1>
                              <1>       ; biosfn_load_text_user_pat (AL,ES,BP,CX,DX,BL,BH)
                              <1>
                              <1>       ; get_font_access();
                              <1>       ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
                              <1>       ; for(i=0;i<CX;i++)
                              <1>       ; {
                              <1>       ;  src = BP + i * BH;
                              <1>       ;  dest = blockaddr + (DX + i) * 32;
                              <1>       ;  memcpyb(0xA000, dest, ES, src, BH);
                              <1>       ; }
                              <1>       ; release_font_access();
                              <1>       ; if(AL>=0x10)
                              <1>       ; {
                              <1>       ; set_scan_lines(BH);
                              <1>       ; }
                              <1>
00002B7B 50                   <1>       push   eax
00002B7C E83C000000           <1>       call   get_font_access
00002B81 28DB                 <1>       sub    bl, bl ; i = 0
                              <1> ltup_1:
00002B83 88D8                 <1>       mov    al, bl
00002B85 F6E7                 <1>       mul    bh
00002B87 0FB7F0               <1>       movzx  esi, ax
00002B8A 01EE                 <1>       add    esi, ebp
00002B8C 88D8                 <1>       mov    al, bl
00002B8E 28E4                 <1>       sub    ah, ah
00002B90 6601D0               <1>       add    ax, dx ; (DX + i)
00002B93 66C1E005             <1>       shl    ax, 5  ; * 32
00002B97 0FB7F8               <1>       movzx  edi, ax
00002B9A 81C700000A00         <1>       add    edi, 0A0000h
00002BA0 51                   <1>       push   ecx
00002BA1 0FB6CF               <1>       movzx  ecx, bh
00002BA4 F3A4                 <1>       rep    movsb
00002BA6 59                   <1>       pop    ecx
00002BA7 FEC3                 <1>       inc    bl
00002BA9 38CB                 <1>       cmp    bl, cl
00002BAB 75D6                 <1>       jne    short ltup_1
                              <1>       ;
00002BAD E840000000           <1>       call   release_font_access
                              <1>       ;
00002BB2 58                   <1>       pop    eax
```

```
                              <1>       ; if(AL>=0x10)
00002BB3 3C10                 <1>       cmp    al, 10h
00002BB5 7205                 <1>       jb     short ltup_2
                              <1>       ; set_scan_lines(BH);
00002BB7 E875000000           <1>       call   set_scan_lines
                              <1> ltup_2:
00002BBC C3                   <1>       retn
                              <1>
                              <1> get_font_access:
                              <1>       ; 09/07/2016
                              <1>       ;
                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
                              <1>       ; vgabios-0.7a (2011)
                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
                              <1>       ; 'vgabios.c', 'get_font_access'
                              <1>
                              <1>       ; get_font_access()
00002BBD 52                   <1>       push   edx
00002BBE 66BAC403             <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
00002BC2 66B80001             <1>       mov    ax, 0100h
00002BC6 66EF                 <1>       out    dx, ax
00002BC8 66B80204             <1>       mov    ax, 0402h
00002BCC 66EF                 <1>       out    dx, ax
00002BCE 66B80407             <1>       mov    ax, 0704h
00002BD2 66EF                 <1>       out    dx, ax
00002BD4 66B80003             <1>       mov    ax, 0300h
00002BD8 66EF                 <1>       out    dx, ax
00002BDA 66BACE03             <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002BDE 66B80402             <1>       mov    ax, 0204h
00002BE2 66EF                 <1>       out    dx, ax
00002BE4 66B80500             <1>       mov    ax, 0005h
00002BE8 66EF                 <1>       out    dx, ax
00002BEA 66B80604             <1>       mov    ax, 0406h
00002BEE 66EF                 <1>       out    dx, ax
00002BF0 5A                   <1>       pop    edx
00002BF1 C3                   <1>       retn
                              <1>
                              <1> release_font_access:
                              <1>       ; 29/07/2016
                              <1>       ; 09/07/2016
                              <1>       ;
                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
                              <1>       ; vgabios-0.7a (2011)
                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
                              <1>       ; 'vgabios.c', 'release_font_access'
                              <1>
00002BF2 66BAC403             <1>       mov    dx, 3C4h ; VGAREG_SEQU_ADDRESS
00002BF6 66B80001             <1>       mov    ax, 0100h
00002BFA 66EF                 <1>       out    dx, ax
00002BFC 66B80203             <1>       mov    ax, 0302h
00002C00 66EF                 <1>       out    dx, ax
00002C02 66B80403             <1>       mov    ax, 0304h
00002C06 66EF                 <1>       out    dx, ax
00002C08 66B80003             <1>       mov    ax, 0300h
00002C0C 66EF                 <1>       out    dx, ax
00002C0E 66BACC03             <1>       mov    dx, 3CCh ; VGAREG_READ_MISC_OUTPUT
00002C12 EC                   <1>       in     al, dx
00002C13 2401                 <1>       and    al, 01h
00002C15 C0E002               <1>       shl    al, 2
00002C18 0C0A                 <1>       or     al, 0Ah
00002C1A 88C4                 <1>       mov    ah, al
00002C1C B006                 <1>       mov    al, 06h
00002C1E 66BACE03             <1>       mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
00002C22 66EF                 <1>       out    dx, ax
00002C24 66B80400             <1>       mov    ax, 0004h
00002C28 66EF                 <1>       out    dx, ax
00002C2A 66B80510             <1>       mov    ax, 1005h
00002C2E 66EF                 <1>       out    dx, ax
00002C30 C3                   <1>       retn
                              <1>
                              <1> set_scan_lines:
                              <1>       ; 09/07/2016
                              <1>       ;
                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
                              <1>       ; vgabios-0.7a (2011)
                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
                              <1>       ; 'vgabios.c', 'set_scan_lines'
                              <1>
                              <1>       ; set_scan_lines(lines)
                              <1>       ; BH = lines
                              <1>
                              <1>       ; outb(crtc_addr, 0x09);
00002C31 66BAD403             <1>       mov    dx, 3D4h ; CRTC_ADDRESS = 3D4h (always)
00002C35 B009                 <1>       mov    al, 09h
00002C37 EE                   <1>       out    dx, al
                              <1>       ; crtc_r9 = inb(crtc_addr+1);
00002C38 6642                 <1>       inc    dx ; 3D5h
00002C3A EC                   <1>       in     al, dx
                              <1>       ; crtc_r9 = (crtc_r9 & 0xe0) | (lines - 1);
00002C3B 24E0                 <1>       and    al, 0E0h
00002C3D FECF                 <1>       dec    bh ; lines - 1
00002C3F 08F8                 <1>       or     al, bh
                              <1>       ; outb(crtc_addr+1, crtc_r9);
00002C41 EE                   <1>       out    dx, al
                              <1>       ;inc   bh
                              <1>       ; if(lines==8)
                              <1>       ;cmp   bh, 8
00002C42 80FF07               <1>       cmp    bh, 7
00002C45 7506                 <1>       jne    short ssl_1
                              <1>       ; biosfn_set_cursor_shape(0x06,0x07);
00002C47 66B90706             <1>       mov    cx, 0607h
00002C4B EB06                 <1>       jmp    short ssl_2
                              <1> ssl_1:
```

```
                             <1>        ; biosfn_set_cursor_shape(lines-4,lines-3);
00002C4D 88F9                <1>        mov    cl, bh ; lines - 1
00002C4F 88CD                <1>        mov    ch, cl ; lines - 1 (16 -> 15)
00002C51 FECD                <1>        dec    ch  ; lines - 2 (16 -> 14)
                             <1> ssl_2:
                             <1>        ; CH = start line, CL = stop line
00002C53 B40A                <1>        mov    ah, 10 ; 6845 register for cursor set
00002C55 66890D[DB5E0000]    <1>        mov    [CURSOR_MODE], cx ; save in data area
00002C5C E812F1FFFF          <1>        call   m16    ; output cx register
                             <1>        ; write_word(BIOSMEM_SEG,BIOSMEM_CHAR_HEIGHT, lines);
00002C61 FEC7                <1>        inc    bh ; lines
00002C63 883D[C65E0000]      <1>        mov    [CHAR_HEIGHT], bh
                             <1>        ;  outb(crtc_addr, 0x12);
00002C69 66BAD403            <1>        mov    dx, 3D4h ; CRTC_ADDRESS
00002C6D B012                <1>        mov    al, 12h
00002C6F EE                  <1>        out    dx, al
                             <1>        ; vde = inb(crtc_addr+1);
00002C70 6642                <1>        inc    dx
00002C72 EC                  <1>        in     al, dx
00002C73 88C4                <1>        mov    ah, al
                             <1>        ; outb(crtc_addr, 0x07);
00002C75 664A                <1>        dec    dx
00002C77 B007                <1>        mov    al, 07h
00002C79 EE                  <1>        out    dx, al
                             <1>        ; ovl = inb(crtc_addr+1);
00002C7A 6642                <1>        inc    dx
00002C7C EC                  <1>        in     al, dx
                             <1>        ; vde += (((ovl & 0x02) << 7) + ((ovl & 0x40) << 3) + 1);
00002C7D 88E2                <1>        mov    dl, ah ; vde
00002C7F 88C6                <1>        mov    dh, al ; ovl
00002C81 6683E002            <1>        and    ax, 02h
00002C85 66C1E007            <1>        shl    ax, 7
00002C89 6689C1              <1>        mov    cx, ax ; (ovl & 0x02) << 7)
00002C8C 88F0                <1>        mov    al, dh ; ovl
00002C8E 6683E040            <1>        and    ax, 40h
00002C92 66C1E003            <1>        shl    ax, 3  ; (ovl & 0x40) << 3)
00002C96 6640                <1>        inc    ax ; + 1
00002C98 6601C8              <1>        add    ax, cx
00002C9B 30F6                <1>        xor    dh, dh
00002C9D 6601D0              <1>        add    ax, dx ; + vde
                             <1>        ; rows = vde / lines;
00002CA0 F6F7                <1>        div    bh
                             <1>        ;dec    al ; rows -1
                             <1>        ; write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, rows-1);
00002CA2 A2[CA5E0000]        <1>        mov    [VGA_ROWS], al ; rows (not 'rows-1' !)
                             <1>        ; write_word(BIOSMEM_SEG,BIOSMEM_PAGE_SIZE, rows * cols * 2);
00002CA7 8A25[C45E0000]      <1>        mov    ah, [CRT_COLS]
00002CAD F6E4                <1>        mul    ah
00002CAF 66D1E0              <1>        shl    ax, 1
00002CB2 66A3[D4650100]      <1>        mov    [CRT_LEN], ax
00002CB8 C3                  <1>        retn
                             <1>
                             <1> load_text_8_14_pat:
                             <1>        ; 26/07/2016
                             <1>        ; 25/07/2016
                             <1>        ; 23/07/2016
                             <1>        ; 09/07/2016
                             <1>        ; load user defined (EGA/VGA) text fonts
                             <1>        ;
                             <1>        ; derived from 'Plex86/Bochs VGABios' source code
                             <1>        ; vgabios-0.7a (2011)
                             <1>        ; by the LGPL VGABios developers Team (2001-2008)
                             <1>        ; 'vgabios.c', 'biosfn_load_text_8_14_pat'
                             <1>
                             <1>        ; biosfn_load_text_8_14_pat (AL,BL)
                             <1>
                             <1>        ; get_font_access();
                             <1>        ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
                             <1>        ; for(i=0;i<0x100;i++)
                             <1>        ; {
                             <1>        ;  src = i * 14;
                             <1>        ;  dest = blockaddr + i * 32;
                             <1>        ;  memcpyb(0xA000, dest, 0xC000, vgafont14+src, 14);
                             <1>        ; }
                             <1>        ; release_font_access();
                             <1>        ; if(AL>=0x10)
                             <1>        ; {
                             <1>        ; set_scan_lines(14);
                             <1>        ; }
                             <1>
00002CB9 50                  <1>        push   eax
00002CBA E8FEFEFFFF          <1>        call   get_font_access
                             <1>
                             <1>        ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
                             <1>        ;mov   dl, bl
                             <1>        ;and   dl, 3
                             <1>        ;shl   dx, 14
                             <1>        ;xchg  dx, bx
                             <1>        ;and   dl, 4
                             <1>        ;shl   dx, 11
                             <1>        ;add   dx, bx
                             <1>
                             <1>        ;xor   dx, dx  ; blockaddr = 0
                             <1>        ; Always block 0 for TRDOS 386 ! (blockaddr=0(
                             <1>
00002CBF 28DB                <1>        sub    bl, bl ; i = 0
00002CC1 B70E                <1>        mov    bh, 14
00002CC3 BE[B8340100]        <1>        mov    esi, vgafont14
00002CC8 BF00000A00          <1>        mov    edi, 0A0000h
                             <1> lt8_14_1:
                             <1>        ;mov   al, bl
                             <1>        ;mul   bh
                             <1>        ;movzx esi, ax
```

```
                              <1>       ;add   esi, vgafont14
                              <1>       ;mov   al, bl
                              <1>       ;sub   ah, ah
                              <1>       ;shl   ax, 5 ; * 32
                              <1>       ;;add  ax, dx ; blockaddr + i * 32;
                              <1>       ;movzx edi, ax ; dest
                              <1>       ;add   edi, 0A0000h
00002CCD 0FB6CF               <1>       movzx  ecx, bh
00002CD0 F3A4                 <1>       rep    movsb
00002CD2 83C712               <1>       add    edi, 18 ; 32 - 14
00002CD5 FEC3                 <1>       inc    bl
00002CD7 75F4                 <1>       jnz    short lt8_14_1
                              <1>       ;
00002CD9 E814FFFFFF           <1>       call   release_font_access
                              <1>       ;
00002CDE 58                   <1>       pop    eax
                              <1>       ; if(AL>=0x10)
00002CDF 3C10                 <1>       cmp    al, 10h
00002CE1 7205                 <1>       jb     short lt8_14_4
                              <1>       ; BH = 14
                              <1>       ; set_scan_lines(14);
00002CE3 E849FFFFFF           <1>       call   set_scan_lines
                              <1> lt8_14_4:
00002CE8 C3                   <1>       retn
                              <1>
                              <1> load_text_8_8_pat:
                              <1>       ; 26/07/2016
                              <1>       ; 25/07/2016
                              <1>       ; 23/07/2016
                              <1>       ; 09/07/2016
                              <1>       ; load user defined (EGA/VGA) text fonts
                              <1>       ;
                              <1>       ; derived from 'Plex86/Bochs VGABios' source code
                              <1>       ; vgabios-0.7a (2011)
                              <1>       ; by the LGPL VGABios developers Team (2001-2008)
                              <1>       ; 'vgabios.c', 'biosfn_load_text_8_8_pat'
                              <1>
                              <1>       ; biosfn_load_text_8_8_pat (AL,BL)
                              <1>
                              <1>       ; get_font_access();
                              <1>       ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
                              <1>       ; for(i=0;i<0x100;i++)
                              <1>       ; {
                              <1>       ;  src = i * 8;
                              <1>       ;  dest = blockaddr + i * 32;
                              <1>       ;  memcpyb(0xA000, dest, 0xC000, vgafont8+src, 8);
                              <1>       ; }
                              <1>       ; release_font_access();
                              <1>       ; if(AL>=0x10)
                              <1>       ; {
                              <1>       ; set_scan_lines(8);
                              <1>       ; }
                              <1>
00002CE9 50                   <1>       push   eax
00002CEA E8CEFEFFFF           <1>       call   get_font_access
                              <1>
                              <1>       ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
                              <1>       ;mov   dl, bl
                              <1>       ;and   dl, 3
                              <1>       ;shl   dx, 14
                              <1>       ;xchg  dx, bx
                              <1>       ;and   dl, 4
                              <1>       ;shl   dx, 11
                              <1>       ;add   dx, bx
                              <1>
                              <1>       ;xor   dx, dx  ; blockaddr = 0
                              <1>       ; Always block 0 for TRDOS 386 ! (blockaddr=0(
                              <1>
00002CEF 28DB                 <1>       sub    bl, bl ; i = 0
00002CF1 B708                 <1>       mov    bh, 8
00002CF3 BE[B82C0100]         <1>       mov    esi, vgafont8
00002CF8 BF00000A00           <1>       mov    edi, 0A0000h
                              <1> lt8_8_1:
                              <1>       ;mov   al, bl
                              <1>       ;mul   bh
                              <1>       ;movzx esi, ax
                              <1>       ;add   esi, vgafont8
                              <1>       ;mov   al, bl
                              <1>       ;sub   ah, ah
                              <1>       ;shl   ax, 5 ; * 32
                              <1>       ;;add  ax, dx ; blockaddr + i * 32;
                              <1>       ;movzx edi, ax ; dest
                              <1>       ;add   edi, 0A0000h
00002CFD 0FB6CF               <1>       movzx  ecx, bh
00002D00 F3A4                 <1>       rep    movsb
00002D02 83C718               <1>       add    edi, 24 ; 32 - 8
00002D05 FEC3                 <1>       inc    bl
00002D07 75F4                 <1>       jnz    short lt8_8_1
                              <1>       ;
00002D09 E8E4FEFFFF           <1>       call   release_font_access
                              <1>       ;
00002D0E 58                   <1>       pop    eax
                              <1>       ; if(AL>=0x10)
00002D0F 3C10                 <1>       cmp    al, 10h
00002D11 7205                 <1>       jb     short lt8_8_2
                              <1>       ; BH = 8
                              <1>       ; set_scan_lines(8);
00002D13 E819FFFFFF           <1>       call   set_scan_lines
                              <1> lt8_8_2:
00002D18 C3                   <1>       retn
                              <1>
                              <1> load_text_8_16_pat:
                              <1>       ; 26/07/2016
```

```
                             <1>        ; 25/07/2016
                             <1>        ; 23/07/2016
                             <1>        ; 09/07/2016
                             <1>        ; load user defined (EGA/VGA) text fonts
                             <1>        ;
                             <1>        ; derived from 'Plex86/Bochs VGABios' source code
                             <1>        ; vgabios-0.7a (2011)
                             <1>        ; by the LGPL VGABios developers Team (2001-2008)
                             <1>        ; 'vgabios.c', 'biosfn_load_text_8_16_pat'
                             <1>
                             <1>        ; biosfn_load_text_8_16_pat (AL,BL)
                             <1>
                             <1>        ; get_font_access();
                             <1>        ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
                             <1>        ; for(i=0;i<0x100;i++)
                             <1>        ; {
                             <1>        ;  src = i * 16;
                             <1>        ;  dest = blockaddr + i * 32;
                             <1>        ;  memcpyb(0xA000, dest, 0xC000, vgafont16+src, 16);
                             <1>        ; }
                             <1>        ; release_font_access();
                             <1>        ; if(AL>=0x10)
                             <1>        ; {
                             <1>        ; set_scan_lines(16);
                             <1>        ; }
                             <1>
00002D19 50                  <1>        push   eax
00002D1A E89EFEFFFF          <1>        call   get_font_access
                             <1>
                             <1>        ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
                             <1>        ;mov   dl, bl
                             <1>        ;and   dl, 3
                             <1>        ;shl   dx, 14
                             <1>        ;xchg  dx, bx
                             <1>        ;and   dl, 4
                             <1>        ;shl   dx, 11
                             <1>        ;add   dx, bx
                             <1>
                             <1>        ;xor   dx, dx  ; blockaddr = 0
                             <1>        ; Always block 0 for TRDOS 386 ! (blockaddr=0(
                             <1>
00002D1F 28DB                <1>        sub    bl, bl ; i = 0
00002D21 B710                <1>        mov    bh, 16
00002D23 BE[B8420100]        <1>        mov    esi, vgafont16
00002D28 BF00000A00          <1>        mov    edi, 0A0000h
00002D2D 0FB6C7              <1>        movzx  eax, bh
                             <1> lt8_16_1:
                             <1>        ;mov   al, bl
                             <1>        ;mul   bh
                             <1>        ;movzx esi, ax
                             <1>        ;add   esi, vgafont16
                             <1>        ;mov   al, bl ; i
                             <1>        ;sub   ah, ah
                             <1>        ;shl   ax, 5 ; * 32
                             <1>        ;;add  ax, dx ; blockaddr + i * 32;
                             <1>        ;movzx edi, ax ; dest
                             <1>        ;add   edi, 0A0000h
                             <1>        ;movzx ecx, bh
00002D30 89C1                <1>        mov    ecx, eax ; 16
00002D32 F3A4                <1>        rep    movsb
00002D34 01C7                <1>        add    edi, eax ; add edi, 16
00002D36 FEC3                <1>        inc    bl
00002D38 75F6                <1>        jnz    short lt8_16_1
                             <1>        ;
00002D3A E8B3FEFFFF          <1>        call   release_font_access
                             <1>        ;
00002D3F 58                  <1>        pop    eax
                             <1>        ; if(AL>=0x10)
00002D40 3C10                <1>        cmp    al, 10h
00002D42 7205                <1>        jb     short lt8_16_2
                             <1>        ; BH = 16
                             <1>        ; set_scan_lines(16);
00002D44 E8E8FEFFFF          <1>        call   set_scan_lines
                             <1> lt8_16_2:
00002D49 C3                  <1>        retn
                             <1>
                             <1> load_gfx_user_chars:
                             <1>        ; 08/08/2016
                             <1>        ; 10/07/2016
                             <1>        ; Setup User-Defined Font for Graphics Mode (VGA)
                             <1>        ;
                             <1>        ; derived from 'Plex86/Bochs VGABios' source code
                             <1>        ; vgabios-0.7a (2011)
                             <1>        ; by the LGPL VGABios developers Team (2001-2008)
                             <1>        ; 'vgabios.c', 'biosfn_load_gfx_user_chars'
                             <1>
                             <1>        ; biosfn_load_gfx_user_chars (ES,BP,CX,BL,DL)
                             <1>        ; /* set 0x43 INT pointer */
                             <1>        ; write_word(0x0, 0x43*4, BP);
                             <1>        ; write_word(0x0, 0x43*4+2, ES);
00002D4A 31C0                <1>        xor    eax, eax
00002D4C 48                  <1>        dec    eax ; 0FFFFFFFFh (user defined fonts)
00002D4D A3[E6650100]        <1>        mov    [VGA_INT43H], eax
                             <1>
                             <1>        ; BL   screen rows code: 00H = user-specified (in DL)
                             <1>          ;                        01H = 14 rows
                             <1>          ;                        02H = 25 rows
                             <1>          ;                        03H = 43 rows
                             <1>          ; CX   bytes per character definition
                             <1>          ; DL   (when BL=0) custom number of character rows on screen
                             <1>        ; dh = 0 -> 256 characters
                             <1>        ; dh = 80h -> 128 characters
                             <1>        ; (If DH <> 0 and DH <> 80h -> invalid)
                             <1>         ; EBP  address of font-definition information (user's mem space)
                             <1> 
                             <1>       ; switch (BL) {
                             <1>       ; case 0:
                             <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, DL-1);
                             <1>       ;    break;
00002D52 20DB                <1>       and    bl, bl
00002D54 7508                <1>       jnz    short l_gfx_uc_1
00002D56 8815[CA5E0000]      <1>       mov    [VGA_ROWS], dl  ; not DL-1 !
00002D5C EB23                <1>       jmp    short l_gfx_uc_4
                             <1> l_gfx_uc_1:
                             <1>       ; case 1:
                             <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 13);
                             <1>       ;    break;
00002D5E FECB                <1>       dec    bl
00002D60 7509                <1>       jnz    short l_gfx_uc_2
                             <1>       ; bl = 1
00002D62 C605[CA5E0000]0E    <1>       mov    byte [VGA_ROWS], 14  ; not 13 !
00002D69 EB16                <1>       jmp    short l_gfx_uc_4
                             <1> l_gfx_uc_2:
00002D6B FECB                <1>       dec    bl
00002D6D 740B                <1>       jz     short l_gfx_uc_3 ; bl = 2
00002D6F FECB                <1>       dec    bl
00002D71 750E                <1>       jnz    short l_gfx_uc_4 ; bl > 3
                             <1>       ; bl = 3
                             <1>       ; case 3:
                             <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 42);
                             <1>       ;    break;
00002D73 C605[CA5E0000]2B    <1>       mov    byte [VGA_ROWS], 43  ; not 42 !
                             <1> l_gfx_uc_3:
                             <1>       ; case 2:
                             <1>       ; default:
                             <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 24);
                             <1>       ;    break;
                             <1>       ; bl = 2 or bl > 3
00002D7A C605[CA5E0000]19    <1>       mov    byte [VGA_ROWS], 25  ; not 24 !
                             <1>       ; }
                             <1> l_gfx_uc_4:
                             <1>       ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, CX);
00002D81 880D[C65E0000]      <1>       mov    [CHAR_HEIGHT], cl
                             <1>       ; }
00002D87 C3                  <1>       retn
                             <1> 
                             <1> load_gfx_8_14_chars:
                             <1>       ; 08/08/2016
                             <1>       ; 10/07/2016
                             <1>       ; Setup ROM 8x14 Font for Graphics Mode (VGA)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'biosfn_load_gfx_8_14_chars'
                             <1> 
                             <1>       ; biosfn_load_gfx_8_14_chars (BL)
                             <1>       ; /* set 0x43 INT pointer */
                             <1>       ; write_word(0x0, 0x43*4, &vgafont14);
                             <1>       ; write_word(0x0, 0x43*4+2, 0xC000);
00002D88 C705[E6650100]-     <1>       mov    dword [VGA_INT43H], vgafont14
00002D8E [B8340100]          <1> 
                             <1> 
                             <1>       ; BL    screen rows code: 00H = user-specified (in DL)
                             <1>         ;                         01H = 14 rows
                             <1>         ;                         02H = 25 rows
                             <1>         ;                         03H = 43 rows
                             <1>         ; DL    (when BL=0) custom number of char rows on screen
                             <1> 
                             <1>       ; switch (BL) {
                             <1>       ; case 0:
                             <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, DL-1);
                             <1>       ;    break;
00002D92 20DB                <1>       and    bl, bl
00002D94 7508                <1>       jnz    short l_gfx_8_14c_1
00002D96 8815[CA5E0000]      <1>       mov    [VGA_ROWS], dl  ; not DL-1 !
00002D9C EB23                <1>       jmp    short l_gfx_8_14c_4
                             <1> l_gfx_8_14c_1:
                             <1>       ; case 1:
                             <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 13);
                             <1>       ;    break;
00002D9E FECB                <1>       dec    bl
00002DA0 7509                <1>       jnz    short l_gfx_8_14c_2
                             <1>       ; bl = 1
00002DA2 C605[CA5E0000]0E    <1>       mov    byte [VGA_ROWS], 14  ; not 13 !
00002DA9 EB16                <1>       jmp    short l_gfx_8_14c_4
                             <1> l_gfx_8_14c_2:
00002DAB FECB                <1>       dec    bl
00002DAD 740B                <1>       jz     short l_gfx_8_14c_3 ; bl = 2
00002DAF FECB                <1>       dec    bl
00002DB1 750E                <1>       jnz    short l_gfx_8_14c_4 ; bl > 3
                             <1>       ; bl = 3
                             <1>       ; case 3:
                             <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 42);
                             <1>       ;    break;
00002DB3 C605[CA5E0000]2B    <1>       mov    byte [VGA_ROWS], 43  ; not 42 !
                             <1> l_gfx_8_14c_3:
                             <1>       ; case 2:
                             <1>       ; default:
                             <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 24);
                             <1>       ;    break;
                             <1>       ; bl = 2 or bl > 3
00002DBA C605[CA5E0000]19    <1>       mov    byte [VGA_ROWS], 25  ; not 24 !
                             <1>       ; }
                             <1> l_gfx_8_14c_4:
                             <1>       ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 14);
```

```
00002DC1 C605[C65E0000]0E    <1>         mov     byte [CHAR_HEIGHT], 14
                             <1>       ; }
00002DC8 C3                  <1>       retn
                             <1> 
                             <1> load_gfx_8_8_chars:
                             <1>       ; 08/08/2016
                             <1>       ; 10/07/2016
                             <1>       ; Setup ROM 8x14 Font for Graphics Mode (VGA)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'biosfn_load_gfx_8_8_dd_chars'
                             <1> 
                             <1>       ; biosfn_load_gfx_8_8_dd_chars (BL)
                             <1>       ; /* set 0x43 INT pointer */
                             <1>       ; write_word(0x0, 0x43*4, &vgafont8);
                             <1>       ; write_word(0x0, 0x43*4+2, 0xC000);
00002DC9 C705[E6650100]-     <1>       mov    dword [VGA_INT43H], vgafont8
00002DCF [B82C0100]          <1> 
                             <1> 
                             <1>       ; BL    screen rows code: 00H = user-specified (in DL)
                             <1>         ;                         01H = 14 rows
                             <1>         ;                         02H = 25 rows
                             <1>         ;                         03H = 43 rows
                             <1>         ; DL    (when BL=0) custom number of char rows on screen
                             <1> 
                             <1>       ; switch (BL) {
                             <1>       ; case 0:
                             <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, DL-1);
                             <1>       ;    break;
00002DD3 20DB                <1>       and    bl, bl
00002DD5 7508                <1>       jnz    short l_gfx_8_8c_1
00002DD7 8815[CA5E0000]      <1>       mov    [VGA_ROWS], dl  ; not DL-1 !
00002DDD EB23                <1>       jmp    short l_gfx_8_8c_4
                             <1> l_gfx_8_8c_1:
                             <1>       ; case 1:
                             <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 13);
                             <1>       ;    break;
00002DDF FECB                <1>       dec    bl
00002DE1 7509                <1>       jnz    short l_gfx_8_8c_2
                             <1>       ; bl = 1
00002DE3 C605[CA5E0000]0E    <1>       mov    byte [VGA_ROWS], 14  ; not 13 !
00002DEA EB16                <1>       jmp    short l_gfx_8_8c_4
                             <1> l_gfx_8_8c_2:
00002DEC FECB                <1>       dec    bl
00002DEE 740B                <1>       jz     short l_gfx_8_8c_3 ; bl = 2
00002DF0 FECB                <1>       dec    bl
00002DF2 750E                <1>       jnz    short l_gfx_8_8c_4 ; bl > 3
                             <1>       ; bl = 3
                             <1>       ; case 3:
                             <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 42);
                             <1>       ;    break;
00002DF4 C605[CA5E0000]2B    <1>       mov    byte [VGA_ROWS], 43  ; not 42 !
                             <1> l_gfx_8_8c_3:
                             <1>       ; case 2:
                             <1>       ; default:
                             <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 24);
                             <1>       ;    break;
                             <1>       ; bl = 2 or bl > 3
00002DFB C605[CA5E0000]19    <1>       mov    byte [VGA_ROWS], 25  ; not 24 !
                             <1>       ; }
                             <1> l_gfx_8_8c_4:
                             <1>       ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 8);
00002E02 C605[C65E0000]08    <1>         mov     byte [CHAR_HEIGHT], 8
                             <1>       ; }
00002E09 C3                  <1>       retn
                             <1> 
                             <1> load_gfx_8_16_chars:
                             <1>       ; 08/08/2016
                             <1>       ; 10/07/2016
                             <1>       ; Setup ROM 8x14 Font for Graphics Mode (VGA)
                             <1>       ;
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios developers Team (2001-2008)
                             <1>       ; 'vgabios.c', 'biosfn_load_gfx_8_16_chars'
                             <1> 
                             <1>       ; biosfn_load_gfx_8_16_chars (BL)
                             <1>       ; /* set 0x43 INT pointer */
                             <1>       ; write_word(0x0, 0x43*4, &vgafont16);
                             <1>       ; write_word(0x0, 0x43*4+2, 0xC000);
00002E0A C705[E6650100]-     <1>       mov    dword [VGA_INT43H], vgafont16
00002E10 [B8420100]          <1> 
                             <1> 
                             <1>       ; BL    screen rows code: 00H = user-specified (in DL)
                             <1>         ;                         01H = 14 rows
                             <1>         ;                         02H = 25 rows
                             <1>         ;                         03H = 43 rows
                             <1>         ; DL    (when BL=0) custom number of char rows on screen
                             <1> 
                             <1>       ; switch (BL) {
                             <1>       ; case 0:
                             <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, DL-1);
                             <1>       ;    break;
00002E14 20DB                <1>       and    bl, bl
00002E16 7508                <1>       jnz    short l_gfx_8_16c_1
00002E18 8815[CA5E0000]      <1>       mov    [VGA_ROWS], dl  ; not DL-1 !
00002E1E EB23                <1>       jmp    short l_gfx_8_16c_4
                             <1> l_gfx_8_16c_1:
                             <1>       ; case 1:
                             <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 13);
                             <1>       ;    break;
```

```
00002E20 FECB                    <1>       dec    bl
00002E22 7509                    <1>       jnz    short l_gfx_8_16c_2
                                 <1>       ; bl = 1
00002E24 C605[CA5E0000]0E        <1>       mov    byte [VGA_ROWS], 14  ; not 13 !
00002E2B EB16                    <1>       jmp    short l_gfx_8_16c_4
                                 <1> l_gfx_8_16c_2:
00002E2D FECB                    <1>       dec    bl
00002E2F 740B                    <1>       jz     short l_gfx_8_16c_3 ; bl = 2
00002E31 FECB                    <1>       dec    bl
00002E33 750E                    <1>       jnz    short l_gfx_8_16c_4 ; bl > 3
                                 <1>       ; bl = 3
                                 <1>       ; case 3:
                                 <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 42);
                                 <1>       ;    break;
00002E35 C605[CA5E0000]2B        <1>       mov    byte [VGA_ROWS], 43  ; not 42 !
                                 <1> l_gfx_8_16c_3:
                                 <1>       ; case 2:
                                 <1>       ; default:
                                 <1>       ;    write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 24);
                                 <1>       ;    break;
                                 <1>       ; bl = 2 or bl > 3
00002E3C C605[CA5E0000]19        <1>       mov    byte [VGA_ROWS], 25  ; not 24 !
                                 <1>       ; }
                                 <1> l_gfx_8_16c_4:
                                 <1>       ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 16);
00002E43 C605[C65E0000]10        <1>         mov     byte [CHAR_HEIGHT], 16
                                 <1>       ; }
00002E4A C3                      <1>       retn
                                 <1>
                                 <1> get_font_info:
                                 <1>       ; 19/09/2016
                                 <1>       ; 08/08/2016
                                 <1>       ; 10/07/2016
                                 <1>       ; Get Current Character Generator Info (VGA)
                                 <1>       ;
                                 <1>       ; derived from 'Plex86/Bochs VGABios' source code
                                 <1>       ; vgabios-0.7a (2011)
                                 <1>       ; by the LGPL VGABios developers Team (2001-2008)
                                 <1>       ; 'vgabios.c', 'biosfn_get_font_info'
                                 <1>
                                 <1>       ; Modified for TRDOS 386 !
                                 <1>       ;
                                 <1>       ; INPUT ->
                                 <1>       ;    AX = 1130h
                                 <1>       ;    BL = 0 -> Get info for current VGA font
                                 <1>       ;             (BH = unused)
                                 <1>       ;    19/09/2016
                                 <1>       ;    BL > 0 -> Get requested character font data
                                 <1>       ;       BL = 1 -> vgafont8
                                 <1>       ;         BL = 2 -> vgafont14
                                 <1>       ;       BL = 3 -> vgafont16
                                 <1>       ;       BL > 3 -> Invalid function (for now!)
                                 <1>       ;       BH = ASCII code of the first character
                                 <1>       ;       ECX = Number of characters from the 1st char
                                 <1>       ;       ECX >= 256 -> All (256-BH) characters
                                 <1>       ;       ECX = 0 -> All characters (BH = unused)
                                 <1>       ;         EDX = User's Buffer Address
                                 <1>       ; OUTPUT ->
                                 <1>       ;    AL = height (scanlines), bytes per character
                                 <1>       ;    AH = screen rows
                                 <1>       ;    Byte 16-23 of EAX = number of columns
                                 <1>       ;    Byte 24-31 of EAX =
                                 <1>       ;        0 -> default font (not configured yet)
                                 <1>       ;        0FFh -> user defined font
                                 <1>       ;        14 = vgafont14
                                 <1>       ;         8 = vgafont8
                                 <1>       ;        16 = vgafont16
                                 <1>       ;   If BL input > 0 ->
                                 <1>       ;       EAX = Actual transfer count
                                 <1>       ;
00002E4B 20DB                    <1>       and    bl, bl
00002E4D 7408                    <1>       jz     short gfi_0
                                 <1>       ; invalid function (input)
00002E4F 80FB03                  <1>       cmp    bl, 3
00002E52 7642                    <1>       jna    short gfi_4
00002E54 31C0                    <1>       xor    eax, eax ; 0
00002E56 C3                      <1>       retn
                                 <1> gfi_0:
00002E57 A0[C65E0000]            <1>       mov    al, [CHAR_HEIGHT]
00002E5C 8A25[CA5E0000]          <1>       mov    ah, [VGA_ROWS]
00002E62 C1E010                  <1>       shl    eax, 16
00002E65 A0[C45E0000]            <1>       mov    al, [CRT_COLS]
00002E6A 8B0D[E6650100]          <1>       mov    ecx, [VGA_INT43H]
00002E70 21C9                    <1>       and    ecx, ecx
00002E72 741E                    <1>       jz     short gfi_2 ; 0 = default font
00002E74 41                      <1>       inc    ecx ; 0FFFFFFFFh -> 0 (user defined font)
00002E75 7504                    <1>       jnz    short gfi_1
00002E77 FECC                    <1>       dec    ah ; 0FFh
00002E79 EB17                    <1>       jmp    short gfi_2
                                 <1> gfi_1:
00002E7B 49                      <1>       dec    ecx ; 08/08/2016
00002E7C B40E                    <1>       mov    ah, 14
00002E7E 81F9[B8340100]          <1>       cmp    ecx, vgafont14
00002E84 740C                    <1>       je     short gfi_2
00002E86 B408                    <1>       mov    ah, 8
00002E88 81F9[B82C0100]          <1>       cmp    ecx, vgafont8
00002E8E 7402                    <1>       je     short gfi_2
                                 <1>       ; vgafont16
00002E90 D0E4                    <1>       shl    ah, 1 ; ah = 16
                                 <1> gfi_2:
00002E92 C1C010                  <1>       rol    eax, 16
                                 <1> gfi_3:
00002E95 C3                      <1>       retn
```

```
                                  <1> gfi_4:
00002E96 89D7                     <1>       mov    edi, edx ; **
00002E98 80FB02                   <1>       cmp    bl, 2
00002E9B 720B                     <1>       jb     short gfi_5
00002E9D 772F                     <1>       ja     short gfi_7
                                  <1>       ;BL = 2 -> vgafont14
00002E9F BE[B8340100]             <1>       mov    esi, vgafont14 ; *
00002EA4 B30E                     <1>       mov    bl, 14
00002EA6 EB07                     <1>       jmp    short gfi_6
                                  <1> gfi_5:
                                  <1>       ;BL = 1 -> vgafont8
00002EA8 BE[B82C0100]             <1>       mov    esi, vgafont8 ; *
00002EAD B308                     <1>       mov    bl, 8
                                  <1> gfi_6:
00002EAF 09C9                     <1>       or     ecx, ecx
00002EB1 7424                     <1>       jz     short gfi_8 ; all chars from the 00h
00002EB3 88F8                     <1>       mov    al, bh ; character index
00002EB5 F6E3                     <1>       mul    bl ; char index * char height/size
00002EB7 0FB7D0                   <1>       movzx  edx, ax
00002EBA 01D6                     <1>       add    esi, edx ; *
00002EBC 66BAFF00                 <1>       mov    dx, 255
00002EC0 28FA                     <1>       sub    dl, bh
00002EC2 6642                     <1>       inc    dx
00002EC4 39D1                     <1>       cmp    ecx, edx
00002EC6 770F                     <1>       ja     short gfi_8
00002EC8 7412                     <1>       je     short gfi_9
00002ECA 89D1                     <1>       mov    ecx, edx
00002ECC EB0E                     <1>       jmp    short gfi_9
                                  <1> gfi_7:
                                  <1>       ;BL = 3 -> vgafont16
00002ECE BE[B8420100]             <1>       mov    esi, vgafont16 ; *
00002ED3 B310                     <1>       mov    bl, 16
00002ED5 EBD8                     <1>       jmp    short gfi_6
                                  <1> gfi_8:
00002ED7 B900010000               <1>       mov    ecx, 256
                                  <1> gfi_9:
00002EDC 6689C8                   <1>       mov    ax, cx ; character count
00002EDF 30FF                     <1>       xor    bh, bh
00002EE1 66F7E3                   <1>       mul    bx ; char count * char height/size
00002EE4 6689C1                   <1>       mov    cx, ax
                                  <1>
                                  <1>       ; ESI = source address in system space
                                  <1>       ; EDI = user's buffer address
                                  <1>       ; ECX = transfer (byte) count
00002EE7 E88DB80000               <1>       call   transfer_to_user_buffer
00002EEC 89C8                     <1>       mov    eax, ecx ; actual transfer count
00002EEE C3                       <1>       retn
                                  <1>
                                  <1> vga_pal_funcs:
                                  <1>       ; 10/08/2016
                                  <1>       ; VGA Palette functions
                                  <1>       ;
                                  <1>       ; derived from 'Plex86/Bochs VGABios' source code
                                  <1>       ; vgabios-0.7a (2011)
                                  <1>       ; by the LGPL VGABios developers Team (2001-2008)
                                  <1>       ; 'vgabios.c', 'vgarom.asm'
                                  <1>
00002EEF 3C00                     <1>       cmp    al, 0
00002EF1 0F848F000000             <1>         je      set_single_palette_reg
                                  <1> vga_palf_1001:
00002EF7 3C01                     <1>       cmp    al, 1
00002EF9 0F84B4000000             <1>         je      set_overscan_border_color
                                  <1> vga_palf_1002:
00002EFF 3C02                     <1>       cmp    al, 2
00002F01 0F84B0000000             <1>         je      set_all_palette_reg
                                  <1> vga_palf_1003:
00002F07 3C03                     <1>       cmp    al, 3
00002F09 0F84E8000000             <1>         je      toggle_intensity
                                  <1> vga_palf_1007:
00002F0F 3C07                     <1>       cmp    al, 7
00002F11 0F840D010000             <1>         je      get_single_palette_reg
00002F17 7266                     <1>       jb     short vga_palf_unknown
                                  <1> vga_palf_1008:
00002F19 3C08                     <1>       cmp    al, 8
00002F1B 0F8437010000             <1>         je      read_overscan_border_color
                                  <1> vga_palf_1009:
00002F21 3C09                     <1>       cmp    al, 9
00002F23 0F8433010000             <1>       je     get_all_palette_reg
                                  <1> vga_palf_1010:
00002F29 3C10                     <1>       cmp    al, 10h
00002F2B 0F8487010000             <1>       je     set_single_dac_reg
00002F31 724C                     <1>       jb     short vga_palf_unknown
                                  <1> vga_palf_1012:
00002F33 3C12                     <1>       cmp    al, 12h
00002F35 0F8498010000             <1>         je      set_all_dac_reg
00002F3B 7242                     <1>       jb     short vga_palf_unknown
                                  <1> vga_palf_1013:
00002F3D 3C13                     <1>       cmp    al, 13h
00002F3F 0F84CC010000             <1>         je      select_video_dac_color_page
                                  <1> vga_palf_1015:
00002F45 3C15                     <1>       cmp    al, 15h
00002F47 0F8412020000             <1>       je     read_single_dac_reg
00002F4D 7230                     <1>       jb     short vga_palf_unknown
                                  <1> vga_palf_1017:
00002F4F 3C17                     <1>       cmp    al, 17h
00002F51 0F8428020000             <1>         je      read_all_dac_reg
00002F57 7226                     <1>       jb     short vga_palf_unknown
                                  <1> vga_palf_1018:
00002F59 3C18                     <1>       cmp    al, 18h
00002F5B 0F845E020000             <1>         je      set_pel_mask
                                  <1> vga_palf_1019:
00002F61 3C19                     <1>       cmp    al, 19h
00002F63 0F8462020000             <1>         je      read_pel_mask
```

```
                              <1> vga_palf_101A:
00002F69 3C1A                 <1>       cmp    al, 1Ah
00002F6B 0F8468020000         <1>         je      read_video_dac_state
                              <1> vga_palf_101B:
00002F71 3C1B                 <1>       cmp    al, 1Bh
                              <1>       ;jne   short vga_palf_unknown
00002F73 770A                 <1>       ja     short vga_palf_unknown
                              <1>
00002F75 E80CF7FFFF           <1>       call   gray_scale_summing
00002F7A E9D5E5FFFF           <1>         jmp     VIDEO_RETURN
                              <1>
                              <1> vga_palf_unknown:
00002F7F 29C0                 <1>       sub    eax, eax ; 0 = invalid function
00002F81 E9D3E5FFFF           <1>         jmp     _video_return
                              <1>
                              <1> set_single_palette_reg:
                              <1>       ; 10/08/2016
                              <1>       ; Set One Palette Register
                              <1>       ; BL = register number to set
                              <1>       ;     (a 4-bit attribute nibble: 00h-0Fh)
                              <1>       ; BH = 6-bit RGB color to display
                              <1>       ;      for that attribute
                              <1>
00002F86 80FB14               <1>       cmp    bl, 14h
                              <1>       ;ja    short no_actl_reg1
00002F89 0F87C5E5FFFF         <1>       ja     VIDEO_RETURN
00002F8F 6650                 <1>       push   ax
00002F91 6652                 <1>       push   dx
00002F93 66BADA03             <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00002F97 EC                   <1>       in     al, dx
00002F98 66BAC003             <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
00002F9C 88D8                 <1>       mov    al, bl
00002F9E EE                   <1>       out    dx, al
00002F9F 88F8                 <1>       mov    al, bh
00002FA1 EE                   <1>       out    dx, al
00002FA2 B020                 <1>       mov    al, 20h
00002FA4 EE                   <1>       out    dx, al
                              <1>       ; ifdef VBOX
00002FA5 66BADA03             <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00002FA9 EC                   <1>       in     al, dx
                              <1>       ; endif ; VBOX
00002FAA 665A                 <1>       pop    dx
00002FAC 6658                 <1>       pop    ax
                              <1> ;no_actl_reg1:
00002FAE E9A1E5FFFF           <1>       jmp    VIDEO_RETURN
                              <1>
                              <1> set_overscan_border_color:
                              <1>       ; 10/08/2016
                              <1>       ; Set Overscan/Border Color Register
                              <1>       ; BH = 6-bit RGB color to display
                              <1>       ;      for that attribute
                              <1>
00002FB3 B311                 <1>       mov    bl, 11h
00002FB5 EBCF                 <1>       jmp    short set_single_palette_reg
                              <1>
                              <1> set_all_palette_reg:
                              <1>       ; 10/08/2016
                              <1>       ; Set All Palette Registers and Overscan
                              <1>       ; EDX = Address of 17 bytes;
                              <1>       ; an rgbRGB value for each of 16 palette
                              <1>       ; registers plus one for the border.
                              <1>
00002FB7 89D6                 <1>       mov    esi, edx ; user buffer
00002FB9 B911000000           <1>       mov    ecx, 17
00002FBE 89E7                 <1>       mov    edi, esp
00002FC0 83EC14               <1>       sub    esp, 20
00002FC3 E8FBB70000           <1>       call   transfer_from_user_buffer
                              <1>       ;jc    VIDEO_RETURN
                              <1>
00002FC8 66BADA03             <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00002FCC EC                   <1>       in     al, dx
00002FCD B100                 <1>       mov    cl, 0
00002FCF 66BAC003             <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
                              <1> set_palette_loop:
00002FD3 88C8                 <1>       mov    al, cl
00002FD5 EE                   <1>       out    dx, al
00002FD6 8A07                 <1>       mov    al, [edi]
00002FD8 EE                   <1>       out    dx, al
00002FD9 47                   <1>       inc    edi
00002FDA FEC1                 <1>       inc    cl
00002FDC 80F910               <1>       cmp    cl, 10h
00002FDF 75F2                 <1>       jne    short set_palette_loop
00002FE1 B011                 <1>       mov    al, 11h
00002FE3 EE                   <1>       out    dx, al
00002FE4 8A07                 <1>       mov    al, [edi]
00002FE6 EE                   <1>       out    dx, al
00002FE7 B020                 <1>       mov    al, 20h
00002FE9 EE                   <1>       out    dx, al
                              <1>       ; ifdef VBOX
00002FEA 66BADA03             <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00002FEE EC                   <1>       in     al, dx
                              <1>       ; endif ; VBOX
00002FEF 83C414               <1>       add    esp, 20
00002FF2 E95DE5FFFF           <1>       jmp    VIDEO_RETURN
                              <1>
                              <1> toggle_intensity:
                              <1>       ; 10/08/2016
                              <1>       ; Select Foreground Blink or Bold Background
                              <1>       ; BL = 00h = enable bold backgrounds
                              <1>       ;          (16 background colors)
                              <1>         ;      01h = enable blinking foreground
                              <1>       ;          (8 background colors)
                              <1>
```

```
00002FF7 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00002FFB EC                  <1>       in     al, dx
00002FFC 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
00003000 B010                <1>       mov    al, 10h
00003002 EE                  <1>       out    dx, al
00003003 66BAC103            <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
00003007 EC                  <1>       in     al, dx
00003008 24F7                <1>       and    al, 0F7h
0000300A 80E301              <1>       and    bl, 01h
0000300D C0E303              <1>       shl    bl, 3
00003010 08D8                <1>       or     al, bl
00003012 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
00003016 EE                  <1>       out    dx, al
00003017 B020                <1>       mov    al, 20h
00003019 EE                  <1>       out    dx, al
                             <1>       ; ifdef VBOX
0000301A 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
0000301E EC                  <1>       in     al, dx
                             <1>       ; endif ; VBOX
0000301F E930E5FFFF          <1>       jmp    VIDEO_RETURN
                             <1>
                             <1> get_single_palette_reg:
                             <1>       ; 10/08/2016
                             <1>       ; Read One Palette Register
                             <1>         ; INPUT:
                             <1>       ; BL = Palette register to read (00h-0Fh)
                             <1>       ; OUTPUT:
                             <1>       ; BH = Current rgbRGB value of specified register
                             <1>       ;      for that attribute
                             <1>
00003024 80FB14              <1>       cmp    bl, 14h
                             <1>       ;ja    short no_actl_reg2
00003027 0F8727E5FFFF        <1>       ja     VIDEO_RETURN
                             <1>
0000302D 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00003031 EC                  <1>       in     al, dx
00003032 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
00003036 88D8                <1>       mov    al, bl
00003038 EE                  <1>       out    dx, al
00003039 66BAC103            <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
0000303D EC                  <1>       in     al, dx
0000303E 8844240D            <1>       mov    [esp+13], al ; bh
00003042 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00003046 EC                  <1>       in     al, dx
00003047 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
0000304B B020                <1>       mov    al, 20h
0000304D EE                  <1>       out    dx, al
                             <1>       ; ifdef VBOX
0000304E 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00003052 EC                  <1>       in     al, dx
                             <1>       ; endif ; VBOX
00003053 E9FCE4FFFF          <1>       jmp    VIDEO_RETURN
                             <1>
                             <1> read_overscan_border_color:
                             <1>       ; 10/08/2016
                             <1>       ; Read Overscan Register
                             <1>       ; OUTPUT:
                             <1>       ; BH = current rgbRGB value
                             <1>       ;      of the overscan/border register
                             <1>
00003058 B311                <1>       mov    bl, 11h
0000305A EBC8                <1>       jmp    short get_single_palette_reg
                             <1>
                             <1> get_all_palette_reg:
                             <1>       ; 10/08/2016
                             <1>       ; Read All Palette Registers
                             <1>       ; EDX = Address of 17-byte buffer
                             <1>       ;      to receive data
                             <1>
0000305C 89D7                <1>       mov    edi, edx
0000305E 89E3                <1>       mov    ebx, esp
00003060 89DE                <1>       mov    esi, ebx
00003062 83EC14              <1>       sub    esp, 20
                             <1>
00003065 B100                <1>       mov    cl, 0
                             <1> get_palette_loop:
00003067 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
0000306B EC                  <1>       in     al, dx
0000306C 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
00003070 88C8                <1>       mov    al, cl
00003072 EE                  <1>       out    dx, al
00003073 66BAC103            <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
00003077 EC                  <1>       in     al, dx
00003078 8803                <1>       mov    [ebx], al
0000307A 43                  <1>       inc    ebx
0000307B FEC1                <1>       inc    cl
0000307D 80F910              <1>       cmp    cl, 10h
00003080 75E5                <1>       jne    short get_palette_loop
00003082 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00003086 EC                  <1>       in     al, dx
00003087 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
0000308B B011                <1>       mov    al, 11h
0000308D EE                  <1>       out    dx, al
0000308E 66BAC103            <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
00003092 EC                  <1>       in     al, dx
00003093 8803                <1>       mov    [ebx], al
00003095 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00003099 EC                  <1>       in     al, dx
0000309A 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
0000309E B020                <1>       mov    al, 20h
000030A0 EE                  <1>       out    dx, al
                             <1>       ; ifdef VBOX
000030A1 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
```

```
000030A5 EC                  <1>       in     al, dx
                             <1>       ; endif ; VBOX
                             <1>
000030A6 B911000000          <1>       mov    ecx, 17 ; transfer (byte) count
                             <1>       ; ESI = source address in system space
                             <1>       ; EDI = user's buffer address
000030AB E8C9B60000          <1>       call   transfer_to_user_buffer
                             <1>
000030B0 83C414              <1>       add    esp, 20
000030B3 E99CE4FFFF          <1>       jmp    VIDEO_RETURN
                             <1>
                             <1> set_single_dac_reg:
                             <1>       ; 10/08/2016
                             <1>       ; Set One DAC Color Register
                             <1>       ; BX = color register to set (0-255)
                             <1>         ; CH = green value (00h-3Fh)
                             <1>         ; CL = blue value  (00h-3Fh)
                             <1>         ; DH = red value   (00h-3Fh)
                             <1>
000030B8 6652                <1>       push   dx
000030BA 66BAC803            <1>       mov    dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
000030BE 88D8                <1>       mov    al, bl
000030C0 EE                  <1>       out    dx, al
                             <1>       ;mov   dx, 3C9h ; VGAREG_DAC_DATA
000030C1 6642                <1>       inc    dx
000030C3 6658                <1>       pop    ax
000030C5 88E0                <1>       mov    al, ah
000030C7 EE                  <1>       out    dx, al
000030C8 88E8                <1>       mov    al, ch
000030CA EE                  <1>       out    dx, al
000030CB 88C8                <1>       mov    al, cl
000030CD EE                  <1>       out    dx, al
000030CE E981E4FFFF          <1>       jmp    VIDEO_RETURN
                             <1>
                             <1> set_all_dac_reg:
                             <1>       ; 12/08/2016
                             <1>       ; 11/08/2016
                             <1>       ; 10/08/2016
                             <1>       ; Set a Block of DAC Color Register
                             <1>       ; BX = first DAC register to set (0-00FFh)
                             <1>       ; ECX = number of registers to set (0-00FFh)
                             <1>       ; EDX = addr of a table of R,G,B values
                             <1>       ;      (it will be CX*3 bytes long)
                             <1>
000030D3 89D6                <1>       mov    esi, edx ; user buffer
000030D5 89CA                <1>       mov    edx, ecx
000030D7 66D1E1              <1>       shl    cx, 1 ; *2
000030DA 01D1                <1>       add    ecx, edx ; ecx = 3*ecx
000030DC 89E5                <1>       mov    ebp, esp
000030DE 89EF                <1>       mov    edi, ebp
000030E0 29CF                <1>       sub    edi, ecx
000030E2 6683E7FC            <1>       and    di, 0FFFCh ; (dword alignment)
000030E6 89FC                <1>       mov    esp, edi
000030E8 E8D6B60000          <1>       call   transfer_from_user_buffer
                             <1>       ;jc    VIDEO_RETURN
                             <1>
000030ED 89D1                <1>       mov    ecx, edx
000030EF 66BAC803            <1>       mov    dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
000030F3 88D8                <1>       mov    al, bl
000030F5 EE                  <1>       out    dx, al
000030F6 66BAC903            <1>       mov    dx, 3C9h ; VGAREG_DAC_DATA
                             <1> set_dac_loop:
000030FA 8A07                <1>       mov    al, [edi]
000030FC EE                  <1>       out    dx, al
000030FD 47                  <1>       inc    edi
000030FE 8A07                <1>       mov    al, [edi]
00003100 EE                  <1>       out    dx, al
00003101 47                  <1>       inc    edi
00003102 8A07                <1>       mov    al, [edi]
00003104 EE                  <1>       out    dx, al
00003105 47                  <1>       inc    edi
00003106 6649                <1>       dec    cx
00003108 75F0                <1>       jnz    short set_dac_loop
0000310A 89EC                <1>       mov    esp, ebp
0000310C E943E4FFFF          <1>       jmp    VIDEO_RETURN
                             <1>
                             <1> select_video_dac_color_page:
                             <1>       ; 10/08/2016
                             <1>       ; DAC Color Paging Functions
                             <1>       ; BL = 00H = select color paging mode
                             <1>         ;       BH = paging mode
                             <1>         ;            00h = 4 blocks of 64 registers
                             <1>         ;            01h = 16 blocks of 16 registers
                             <1>       ; BL = 01H = activate color page
                             <1>         ;       BH = DAC color page number
                             <1>         ;            00h-03h (4-page/64-reg mode)
                             <1>         ;            00h-0Fh (16-page/16-reg mode)
                             <1>
00003111 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00003115 EC                  <1>       in     al, dx
00003116 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
0000311A B010                <1>       mov    al, 10h
0000311C EE                  <1>       out    dx, al
0000311D 66BAC103            <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
00003121 EC                  <1>       in     al, dx
00003122 80E301              <1>       and    bl, 01h
00003125 750E                <1>       jnz    short set_dac_page
00003127 247F                <1>       and    al, 07Fh
00003129 C0E707              <1>       shl    bh, 7
0000312C 08F8                <1>       or     al, bh
0000312E 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
00003132 EE                  <1>       out    dx, al
00003133 EB1D                <1>       jmp    short set_actl_normal
```

```
                             <1> set_dac_page:
00003135 6650                <1>       push   ax
00003137 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
0000313B EC                  <1>       in     al, dx
0000313C 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
00003140 B014                <1>       mov    al, 14h
00003142 EE                  <1>       out    dx, al
00003143 6658                <1>       pop    ax
00003145 2480                <1>       and    al, 80h
00003147 7503                <1>       jnz    short set_dac_16_page
00003149 C0E702              <1>       shl    bh, 2
                             <1> set_dac_16_page:
0000314C 80E70F              <1>       and    bh, 0Fh
0000314F 88F8                <1>       mov    al, bh
00003151 EE                  <1>       out    dx, al
                             <1> set_actl_normal:
00003152 B020                <1>       mov    al, 20h
00003154 EE                  <1>       out    dx, al
                             <1>       ; ifdef VBOX
00003155 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00003159 EC                  <1>       in     al, dx
                             <1>       ; endif ; VBOX
0000315A E9F5E3FFFF          <1>       jmp    VIDEO_RETURN
                             <1>
                             <1> read_single_dac_reg:
                             <1>       ; 10/08/2016
                             <1>       ; Read One DAC Color Register
                             <1>       ; INPUT:
                             <1>       ; BX = color register to read (0-255)
                             <1>       ; OUTPUT:
                             <1>       ; CH = green value (00h-3Fh)
                             <1>         ; CL = blue value  (00h-3Fh)
                             <1>         ; DH = red value   (00h-3Fh)
                             <1>
0000315F 66BAC703            <1>       mov    dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
00003163 88D8                <1>       mov    al, bl
00003165 EE                  <1>       out    dx, al
00003166 66BAC903            <1>       mov    dx, 3C9h ; VGAREG_DAC_DATA
0000316A EC                  <1>       in     al, dx
0000316B 88442415            <1>       mov    [esp+21], al ; dh
0000316F EC                  <1>       in     al, dx
00003170 88C5                <1>       mov    ch, al
00003172 EC                  <1>       in     al, dx
00003173 88C1                <1>       mov    cl, al
00003175 66894C2410          <1>       mov    [esp+16], cx ; cx
0000317A E9D5E3FFFF          <1>       jmp    VIDEO_RETURN
                             <1>
                             <1> read_all_dac_reg:
                             <1>       ; 12/08/2016
                             <1>       ; 11/08/2016
                             <1>       ; 10/08/2016
                             <1>       ; Read a Block of DAC Color Registers
                             <1>         ; BX = first DAC register to read (0-00FFh)
                             <1>         ; ECX = number of registers to read (0-00FFh)
                             <1>         ; EDX = addr of a buffer to hold R,G,B values
                             <1>       ;      (CX*3 bytes long)
                             <1>
0000317F 89D7                <1>       mov    edi, edx ; user buffer
00003181 89CA                <1>       mov    edx, ecx
00003183 66D1E2              <1>       shl    dx, 1 ; *2
00003186 01CA                <1>       add    edx, ecx ; edx = 3*ecx
00003188 89E5                <1>       mov    ebp, esp
0000318A 89EE                <1>       mov    esi, ebp
0000318C 29D6                <1>       sub    esi, edx
0000318E 6683E6FC            <1>       and    si, 0FFFCh ; (dword alignment)
00003192 89F4                <1>       mov    esp, esi
00003194 52                  <1>       push   edx ; 3*ecx
00003195 66BAC703            <1>       mov    dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
00003199 88D8                <1>       mov    al, bl
0000319B EE                  <1>       out    dx, al
0000319C 66BAC903            <1>       mov    dx, 3C9h ; VGAREG_DAC_DATA
000031A0 89F3                <1>       mov    ebx, esi
                             <1> read_dac_loop:
000031A2 EC                  <1>       in     al, dx
000031A3 8803                <1>       mov    [ebx], al
000031A5 43                  <1>       inc    ebx
000031A6 EC                  <1>       in     al, dx
000031A7 8803                <1>       mov    [ebx], al
000031A9 43                  <1>       inc    ebx
000031AA EC                  <1>       in     al, dx
000031AB 8803                <1>       mov    [ebx], al
000031AD 43                  <1>       inc    ebx
000031AE 6649                <1>       dec    cx
000031B0 75F0                <1>       jnz    short read_dac_loop
000031B2 59                  <1>       pop    ecx ; 3*ecx
                             <1>       ; ECX = transfer (byte) count
                             <1>       ; ESI = source address in system space
                             <1>       ; EDI = user's buffer address
000031B3 E8C1B50000          <1>       call   transfer_to_user_buffer
000031B8 89EC                <1>       mov    esp, ebp
000031BA E995E3FFFF          <1>       jmp    VIDEO_RETURN
                             <1>
                             <1> set_pel_mask:
                             <1>       ; 10/08/2016
                             <1>       ; BL = mask value
000031BF 66BAC603            <1>       mov    dx, 3C6h ; VGAREG_PEL_MASK
000031C3 88D8                <1>       mov    al, bl
000031C5 EE                  <1>       out    dx, al
000031C6 E989E3FFFF          <1>       jmp    VIDEO_RETURN
                             <1>
                             <1> read_pel_mask:
                             <1>       ; 10/08/2016
                             <1>       ; Output: BL = mask value
```

```
000031CB 66BAC603            <1>       mov    dx, 3C6h ; VGAREG_PEL_MASK
000031CF EC                  <1>       in     al, dx
000031D0 8844240C            <1>       mov    [esp+12], al ; bl
000031D4 E97BE3FFFF          <1>       jmp    VIDEO_RETURN
                             <1> 
                             <1> read_video_dac_state:
                             <1>       ; 10/08/2016
                             <1>       ; Query DAC Color Paging State
                             <1>       ; Output:
                             <1>       ; BH = current active DAC color page
                             <1>         ; BL = current active DAC paging mode
                             <1> 
000031D9 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
000031DD EC                  <1>       in     al, dx
000031DE 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
000031E2 B010                <1>       mov    al, 10h
000031E4 EE                  <1>       out    dx, al
000031E5 66BAC103            <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
000031E9 EC                  <1>       in     al, dx
000031EA 88C3                <1>       mov    bl, al
000031EC C0EB07              <1>       shr    bl, 7
000031EF 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
000031F3 EC                  <1>       in     al, dx
000031F4 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
000031F8 B014                <1>       mov    al, 14h
000031FA EE                  <1>       out    dx, al
000031FB 66BAC103            <1>       mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
000031FF EC                  <1>       in     al, dx
00003200 88C7                <1>       mov    bh, al
00003202 80E70F              <1>       and    bh, 0Fh
00003205 F6C301              <1>       test   bl, 01
00003208 7503                <1>       jnz    short get_dac_16_page
0000320A C0EF02              <1>       shr    bh, 2
                             <1> get_dac_16_page:
0000320D 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
00003211 EC                  <1>       in     al, dx
00003212 66BAC003            <1>       mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
00003216 B020                <1>       mov    al, 20h
00003218 EE                  <1>       out    dx, al
                             <1>       ; ifdef VBOX
00003219 66BADA03            <1>       mov    dx, 3DAh ; VGAREG_ACTL_RESET
0000321D EC                  <1>       in     al, dx
                             <1>       ; endif ; VBOX
0000321E 66895C240C          <1>       mov    [esp+12], bx ; bx
00003223 E92CE3FFFF          <1>       jmp    VIDEO_RETURN
                             <1> 
                             <1> ; % include 'vidata.s' ; VIDEO DATA
                             <1> 
                             <1> ; /// End Of VIDEO FUNCTIONS ///

                                 setup_rtc_int:
                                 ; source: http://wiki.osdev.org/RTC
00003228 FA                            cli          ; disable interrupts
                                       ; default int frequency is 1024 Hz (Lower 4 bits of register A is 0110b or 6)
                                       ; in order to change this ...
                                       ; frequency  = 32768 >> (rate-1) --> 32768 >> 5 = 1024
                                       ; (rate must be above 2 and not over 15)
                                       ; new rate = 15 --> 32768 >> (15-1) = 2 Hz
00003229 B08A                          mov    al, 8Ah
0000322B E670                          out    70h, al ; set index to register A, disable NMI
0000322D 90                            nop
0000322E E471                          in     al, 71h ; get initial value of register A
00003230 88C4                          mov    ah, al
00003232 80E4F0                        and    ah, 0F0h
00003235 B08A                          mov    al, 8Ah
00003237 E670                          out    70h, al ; reset index to register A
00003239 88E0                          mov    al, ah
0000323B 0C0F                          or     al, 0Fh      ; new rate (0Fh -> 15)
0000323D E671                          out    71h, al ; write only our rate to A. Note, rate is the bottom 4 bits.
                                       ; enable RTC interrupt
0000323F B08B                          mov    al, 8Bh ;
00003241 E670                          out    70h, al ; select register B and disable NMI
00003243 90                            nop
00003244 E471                          in     al, 71h ; read the current value of register B
00003246 88C4                          mov    ah, al  ;
00003248 B08B                          mov    al, 8Bh ;
0000324A E670                          out    70h, al ; set the index again (a read will reset the index to register B)
0000324C 88E0                          mov    al, ah  ;
0000324E 0C40                          or     al, 40h ;
00003250 E671                          out    71h, al ; write the previous value ORed with 0x40. This turns on bit 6 of register B
00003252 FB                            sti
00003253 C3                            retn

                                 ; Write memory information
                                 ; 29/01/2016
                                 ; 06/11/2014
                                 ; 14/08/2015
                                 memory_info:
00003254 A1[3C580100]                  mov    eax, [memory_size] ; in pages
00003259 50                            push   eax
0000325A C1E00C                        shl    eax, 12                ; in bytes
0000325D BB0A000000                    mov    ebx, 10
00003262 89D9                          mov    ecx, ebx        ; 10
00003264 BE[C9180100]                  mov    esi, mem_total_b_str
00003269 E8BD000000                    call   bintdstr
0000326E 58                            pop    eax
0000326F B107                          mov    cl, 7
00003271 BE[ED180100]                  mov    esi, mem_total_p_str
00003276 E8B0000000                    call   bintdstr
                                       ; 14/08/2015
0000327B E8C8000000                    call   calc_free_mem
                                       ; edx = calculated free pages
                                       ; ecx = 0
```

```
00003280 A1[40580100]                  mov    eax, [free_pages]
00003285 39D0                          cmp    eax, edx ; calculated free mem value
                                              ; and initial free mem value are same or not?
00003287 751D                          jne    short pmim ; print mem info with '?' if not
00003289 52                            push   edx ; free memory in pages
                                       ;mov   eax, edx
0000328A C1E00C                        shl    eax, 12 ; convert page count
                                                    ; to byte count
0000328D B10A                          mov    cl, 10
0000328F BE[0D190100]                  mov    esi, free_mem_b_str
00003294 E892000000                    call   bintdstr
00003299 58                            pop    eax
0000329A B107                          mov    cl, 7
0000329C BE[31190100]                  mov    esi, free_mem_p_str
000032A1 E885000000                    call   bintdstr
                                 pmim:
000032A6 BE[B7180100]                  mov    esi, msg_memory_info
                                       ;
000032AB B407                          mov    ah, 07h ; Black background,
                                                    ; light gray forecolor
                                 print_kmsg: ; 29/01/2016
000032AD 8825[67580100]                mov    [ccolor], ah
                                 pkmsg_loop:
000032B3 AC                            lodsb
000032B4 08C0                          or     al, al
000032B6 7410                          jz     short pkmsg_ok
000032B8 56                            push   esi
                                       ; 13/05/2016
000032B9 0FB61D[67580100]              movzx  ebx, byte [ccolor]
                                                    ; Video page 0 (bh=0)
000032C0 E8EDE9FFFF                    call   _write_tty
000032C5 5E                            pop    esi
000032C6 EBEB                          jmp    short pkmsg_loop
                                 pkmsg_ok:
000032C8 C3                            retn

                                 ; Convert binary number to hexadecimal string
                                 ; 10/05/2015
                                 ; dsectpm.s (28/02/2015)
                                 ; Retro UNIX 386 v1 - Kernel v0.2.0.6
                                 ; 01/12/2014
                                 ; 25/11/2014
                                 ;
                                 bytetohex:
                                       ; INPUT ->
                                       ;      AL = byte (binary number)
                                       ; OUTPUT ->
                                       ;      AX = hexadecimal string
                                       ;
000032C9 53                            push   ebx
000032CA 31DB                          xor    ebx, ebx
000032CC 88C3                          mov    bl, al
000032CE C0EB04                        shr    bl, 4
000032D1 8A9B[1B330000]                mov    bl, [ebx+hexchrs]
000032D7 86D8                          xchg   bl, al
000032D9 80E30F                        and    bl, 0Fh
000032DC 8AA3[1B330000]                mov    ah, [ebx+hexchrs]
000032E2 5B                            pop    ebx
000032E3 C3                            retn

                                 wordtohex:
                                       ; INPUT ->
                                       ;      AX = word (binary number)
                                       ; OUTPUT ->
                                       ;      EAX = hexadecimal string
                                       ;
000032E4 53                            push   ebx
000032E5 31DB                          xor    ebx, ebx
000032E7 86E0                          xchg   ah, al
000032E9 6650                          push   ax
000032EB 88E3                          mov    bl, ah
000032ED C0EB04                        shr    bl, 4
000032F0 8A83[1B330000]                mov    al, [ebx+hexchrs]
000032F6 88E3                          mov    bl, ah
000032F8 80E30F                        and    bl, 0Fh
000032FB 8AA3[1B330000]                mov    ah, [ebx+hexchrs]
00003301 C1E010                        shl    eax, 16
00003304 6658                          pop    ax
00003306 5B                            pop    ebx
00003307 EBC0                          jmp    short bytetohex
                                       ;mov   bl, al
                                       ;shr   bl, 4
                                       ;mov   bl, [ebx+hexchrs]
                                       ;xchg  bl, al
                                       ;and   bl, 0Fh
                                       ;mov   ah, [ebx+hexchrs]
                                       ;pop   ebx
                                       ;retn

                                 dwordtohex:
                                       ; INPUT ->
                                       ;      EAX = dword (binary number)
                                       ; OUTPUT ->
                                       ;      EDX:EAX = hexadecimal string
                                       ;
00003309 50                            push   eax
0000330A C1E810                        shr    eax, 16
0000330D E8D2FFFFFF                    call   wordtohex
00003312 89C2                          mov    edx, eax
00003314 58                            pop    eax
00003315 E8CAFFFFFF                    call   wordtohex
0000331A C3                            retn
```

```
                                        ; 10/05/2015
                                        hex_digits:
                                        hexchrs:
0000331B 303132333435363738-                  db '0123456789ABCDEF'
00003324 39414243444546
                                        
                                        ; Convert binary number to decimal/numeric string
                                        ; 06/11/2014
                                        ; Temporary Code
                                        ;
                                        
                                        bintdstr:
                                              ; EAX = binary number
                                              ; ESI = decimal/numeric string address
                                              ; EBX = divisor (10)
                                              ; ECX = string length (<=10)
0000332B 01CE                                 add    esi, ecx
                                        btdstr0:
0000332D 4E                                   dec    esi
0000332E 31D2                                 xor    edx, edx
00003330 F7F3                                 div    ebx
00003332 80C230                               add    dl, 30h
00003335 8816                                 mov    [esi], dl
00003337 FEC9                                 dec    cl
00003339 740C                                 jz     short btdstr2 ; 08/09/2016
0000333B 09C0                                 or     eax, eax
0000333D 75EE                                 jnz    short btdstr0
                                        btdstr1:
0000333F 4E                                   dec    esi
00003340 C60620                                 mov     byte [esi], 20h ; blank space
00003343 FEC9                                 dec    cl
00003345 75F8                                 jnz    short btdstr1
                                        btdstr2:
00003347 C3                                   retn
                                        
                                        ; Calculate free memory pages on M.A.T.
                                        ; 06/11/2014
                                        ; Temporary Code
                                        ;
                                        
                                        calc_free_mem:
00003348 31D2                                 xor    edx, edx
                                              ;xor   ecx, ecx
0000334A 668B0D[50580100]                     mov    cx, [mat_size] ; in pages
00003351 C1E10A                               shl    ecx, 10      ; 1024 dwords per page
00003354 BE00001000                           mov    esi, MEM_ALLOC_TBL
                                        cfm0:
00003359 AD                                   lodsd
0000335A 51                                   push   ecx
0000335B B920000000                           mov    ecx, 32
                                        cfm1:
00003360 D1E8                                 shr    eax, 1
00003362 7301                                 jnc    short cfm2
00003364 42                                   inc    edx
                                        cfm2:
00003365 E2F9                                 loop   cfm1
00003367 59                                   pop    ecx
00003368 E2EF                                 loop   cfm0
0000336A C3                                   retn
                                        
                                        %include 'diskio.s'  ; 07/03/2015
                                    <1> ; ****************************************************************************
                                    <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - diskio.s
                                    <1> ; ----------------------------------------------------------------------------
                                    <1> ; Last Update: 09/12/2017
                                    <1> ; ----------------------------------------------------------------------------
                                    <1> ; Beginning: 24/01/2016
                                    <1> ; ----------------------------------------------------------------------------
                                    <1> ; Assembler: NASM version 2.11 (trdos386.s)
                                    <1> ; ----------------------------------------------------------------------------
                                    <1> ; Turkish Rational DOS
                                    <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
                                    <1> ;
                                    <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                                    <1> ; diskio.inc (22/08/2015)
                                    <1> ;
                                    <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)
                                    <1> ; ****************************************************************************
                                    <1> 
                                    <1> ; Retro UNIX 386 v1 Kernel - DISKIO.INC
                                    <1> ; Last Modification: 22/08/2015
                                    <1> ;     (Initialized Disk Parameters Data is in 'DISKDATA.INC')
                                    <1> ;     (Uninitialized Disk Parameters Data is in 'DISKBSS.INC')
                                    <1> 
                                    <1> ; DISK I/O SYSTEM - Erdogan Tan (Retro UNIX 386 v1 project)
                                    <1> 
                                    <1> ; ///////// DISK I/O SYSTEM ///////////////
                                    <1> 
                                    <1> ; 06/02/2015
                                    <1> diskette_io:
0000336B 9C                         <1>       pushfd
0000336C 0E                         <1>       push   cs
0000336D E809000000                 <1>       call   DISKETTE_IO_1
00003372 C3                         <1>       retn
                                    <1> 
                                    <1> ;;;;;; DISKETTE I/O ;;;;;;;;;;;;;;;;;;; 06/02/2015 ;;;
                                    <1> ;//////////////////////////////////////////////////////
                                    <1> 
                                    <1> ; DISKETTE I/O - Erdogan Tan (Retro UNIX 386 v1 project)
                                    <1> ; 20/02/2015
                                    <1> ; 06/02/2015 (unix386.s)
                                    <1> ; 16/12/2014 - 02/01/2015 (dsectrm2.s)
                                    <1> ;
```

```
<1> ; Code (DELAY) modifications - AWARD BIOS 1999 (ADISK.EQU, COMMON.MAC)
<1> ;
<1> ; ADISK.EQU
<1>
<1> ;----- Wait control constants
<1>
<1> ;amount of time to wait while RESET is active.
<1>
<1> WAITCPU_RESET_ON   EQU    21           ;Reset on must last at least 14us
<1>                                        ;at 250 KBS xfer rate.
<1>                                        ;see INTEL MCS, 1985, pg. 5-456
<1>
<1> WAITCPU_FOR_STATUS EQU    100          ;allow 30 microseconds for
<1>                                        ;status register to become valid
<1>                                        ;before re-reading.
<1>
<1> ;After sending a byte to NEC, status register may remain
<1> ;incorrectly set for 24 us.
<1>
<1> WAITCPU_RQM_LOW           EQU    24           ;number of loops to check for
<1>                                        ;RQM low.
<1>
<1> ; COMMON.MAC
<1> ;
<1> ;     Timing macros
<1> ;
<1>
<1> %macro             SIODELAY 0                ; SHORT IODELAY
<1>              jmp short $+2
<1> %endmacro
<1>
<1> %macro             IODELAY  0                ; NORMAL IODELAY
<1>              jmp short $+2
<1>              jmp short $+2
<1> %endmacro
<1>
<1> %macro             NEWIODELAY 0
<1>              out    0ebh,al
<1> %endmacro
<1>
<1> ; (According to) AWARD BIOS 1999 - ATORGS.ASM (dw -> equ, db -> equ)
<1> ;;; WAIT_FOR_MEM
<1> ;WAIT_FDU_INT_LO   equ    017798       ; 2.5 secs in 30 micro units.
<1> ;WAIT_FDU_INT_HI   equ    1
<1> WAIT_FDU_INT_LH           equ    83334        ; 27/02/2015 (2.5 seconds waiting)
<1> ;;; WAIT_FOR_PORT
<1> ;WAIT_FDU_SEND_LO  equ    16667        ; .5 secons in 30 us units.
<1> ;WAIT_FDU_SEND_HI  equ    0
<1> WAIT_FDU_SEND_LH   equ    16667        ; 27/02/2015
<1> ;Time to wait while waiting for each byte of NEC results = .5
<1> ;seconds.  .5 seconds = 500,000 micros.  500,000/30 = 16,667.
<1> ;WAIT_FDU_RESULTS_LO      equ    16667        ; .5 seconds in 30 micro units.
<1> ;WAIT_FDU_RESULTS_HI      equ    0
<1> WAIT_FDU_RESULTS_LH       equ    16667  ; 27/02/2015
<1> ;;; WAIT_REFRESH
<1> ;amount of time to wait for head settle, per unit in parameter
<1> ;table = 1 ms.
<1> WAIT_FDU_HEAD_SETTLE      equ    33           ; 1 ms in 30 micro units.
<1>
<1>
<1> ; //////////////// DISKETTE I/O ////////////////
<1>
<1> ; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - POSTEQU.INC)
<1>
<1> ;----------------------------------------
<1> ;     EQUATES USED BY POST AND BIOS    :
<1> ;----------------------------------------
<1>
<1> ;--------- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS ------------
<1> ;PORT_A            EQU    060H         ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
<1> ;PORT_B            EQU    061H         ; PORT B READ/WRITE DIAGNOSTIC REGISTER
<1> ;REFRESH_BIT EQU    00010000B    ; REFRESH TEST BIT
<1>
<1> ;----------------------------------------
<1> ;     CMOS EQUATES FOR THIS SYSTEM     :
<1> ;-------------------------------------------------------------------------------
<1> ;CMOS_PORT   EQU    070H         ; I/O ADDRESS OF CMOS ADDRESS PORT
<1> ;CMOS_DATA   EQU    071H         ; I/O ADDRESS OF CMOS DATA PORT
<1> ;NMI         EQU    10000000B    ; DISABLE NMI INTERRUPTS MASK -
<1>                                  ;  HIGH BIT OF CMOS LOCATION ADDRESS
<1>
<1> ;---------- CMOS TABLE LOCATION ADDRESS'S ## ----------------------------------
<1> CMOS_DISKETTE      EQU    010H         ; DISKETTE DRIVE TYPE BYTE      ;
<1> ;            EQU    011H         ; - RESERVED                    ;C
<1> CMOS_DISK    EQU    012H         ; FIXED DISK TYPE BYTE                 ;H
<1> ;            EQU    013H         ; - RESERVED                    ;E
<1> CMOS_EQUIP   EQU    014H         ; EQUIPMENT WORD LOW BYTE       ;C
<1>
<1> ;---------- DISKETTE EQUATES ---------------------------------------------------
<1> INT_FLAG     EQU    10000000B    ; INTERRUPT OCCURRENCE FLAG
<1> DSK_CHG      EQU    10000000B    ; DISKETTE CHANGE FLAG MASK BIT
<1> DETERMINED   EQU    00010000B    ; SET STATE DETERMINED IN STATE BITS
<1> HOME         EQU    00010000B    ; TRACK 0 MASK
<1> SENSE_DRV_ST EQU    00000100B    ; SENSE DRIVE STATUS COMMAND
<1> TRK_SLAP     EQU    030H         ; CRASH STOP (48 TPI DRIVES)
<1> QUIET_SEEK   EQU    00AH         ; SEEK TO TRACK 10
<1> ;MAX_DRV     EQU    2            ; MAX NUMBER OF DRIVES
<1> HD12_SETTLE  EQU    15           ; 1.2 M HEAD SETTLE TIME
<1> HD320_SETTLE EQU    20           ; 320 K HEAD SETTLE TIME
<1> MOTOR_WAIT   EQU    37           ; 2 SECONDS OF COUNTS FOR MOTOR TURN OFF
<1>
<1> ;---------- DISKETTE ERRORS -----------------------------------------------------
<1> ;TIME_OUT    EQU    080H         ; ATTACHMENT FAILED TO RESPOND
```

```
<1> ;BAD_SEEK    EQU    040H         ; SEEK OPERATION FAILED
<1> BAD_NEC      EQU    020H         ; DISKETTE CONTROLLER HAS FAILED
<1> BAD_CRC      EQU    010H         ; BAD CRC ON DISKETTE READ
<1> MED_NOT_FND  EQU    00CH         ; MEDIA TYPE NOT FOUND
<1> DMA_BOUNDARY EQU    009H         ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
<1> BAD_DMA      EQU    008H         ; DMA OVERRUN ON OPERATION
<1> MEDIA_CHANGE EQU    006H         ; MEDIA REMOVED ON DUAL ATTACH CARD
<1> RECORD_NOT_FND      EQU    004H         ; REQUESTED SECTOR NOT FOUND
<1> WRITE_PROTECT       EQU    003H         ; WRITE ATTEMPTED ON WRITE PROTECT DISK
<1> BAD_ADDR_MARK       EQU    002H         ; ADDRESS MARK NOT FOUND
<1> BAD_CMD      EQU    001H         ; BAD COMMAND PASSED TO DISKETTE I/O
<1>
<1> ;---------- DISK CHANGE LINE EQUATES ----------------------------------------------
<1> NOCHGLN      EQU    001H         ; NO DISK CHANGE LINE AVAILABLE
<1> CHGLN        EQU    002H         ; DISK CHANGE LINE AVAILABLE
<1>
<1> ;---------- MEDIA/DRIVE STATE INDICATORS ------------------------------------------
<1> TRK_CAPA     EQU    00000001B    ; 80 TRACK CAPABILITY
<1> FMT_CAPA     EQU    00000010B    ; MULTIPLE FORMAT CAPABILITY (1.2M)
<1> DRV_DET      EQU    00000100B    ; DRIVE DETERMINED
<1> MED_DET      EQU    00010000B    ; MEDIA DETERMINED BIT
<1> DBL_STEP     EQU    00100000B    ; DOUBLE STEP BIT
<1> RATE_MSK     EQU    11000000B    ; MASK FOR CLEARING ALL BUT RATE
<1> RATE_500     EQU    00000000B    ; 500 KBS DATA RATE
<1> RATE_300     EQU    01000000B    ; 300 KBS DATA RATE
<1> RATE_250     EQU    10000000B    ; 250 KBS DATA RATE
<1> STRT_MSK     EQU    00001100B    ; OPERATION START RATE MASK
<1> SEND_MSK     EQU    11000000B    ; MASK FOR SEND RATE BITS
<1>
<1> ;---------- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY ----------------------------
<1> M3D3U        EQU    00000000B    ; 360 MEDIA/DRIVE NOT ESTABLISHED
<1> M3D1U        EQU    00000001B    ; 360 MEDIA,1.2DRIVE NOT ESTABLISHED
<1> M1D1U        EQU    00000010B    ; 1.2 MEDIA/DRIVE NOT ESTABLISHED
<1> MED_UNK      EQU    00000111B    ; NONE OF THE ABOVE
<1>
<1> ;---------- INTERRUPT EQUATES -----------------------------------------------------
<1> ;EOI         EQU    020H         ; END OF INTERRUPT COMMAND TO 8259
<1> ;INTA00             EQU    020H         ; 8259 PORT
<1> INTA01              EQU    021H         ; 8259 PORT
<1> INTB00              EQU    0A0H         ; 2ND 8259
<1> INTB01              EQU    0A1H         ;
<1>
<1> ;----------------------------------------------------------------------------------
<1> DMA08        EQU    008H         ; DMA STATUS REGISTER PORT ADDRESS
<1> DMA          EQU    000H         ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS
<1> DMA18        EQU    0D0H         ; 2ND DMA STATUS PORT ADDRESS
<1> DMA1         EQU    0C0H         ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
<1> ;----------------------------------------------------------------------------------
<1> ;TIMER              EQU    040H         ; 8254 TIMER - BASE ADDRESS
<1>
<1> ;----------------------------------------------------------------------------------
<1> DMA_PAGE     EQU    081H         ; START OF DMA PAGE REGISTERS
<1>
<1> ; 06/02/2015 (unix386.s, protected mode modifications)
<1> ; (unix386.s <-- dsectrm2.s)
<1> ; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - DSEG.INC)
<1>
<1> ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
<1> ; 10/12/2014
<1> ;
<1> ;int40h:
<1> ;     pushf
<1> ;     push   cs
<1> ;     ;cli
<1> ;     call   DISKETTE_IO_1
<1> ;     retn
<1>
<1> ; DSKETTE ----- 04/21/86 DISKETTE BIOS
<1> ; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)
<1> ;
<1>
<1> ;-- INT13H ------------------------------------------------------------------------
<1> ; DISKETTE I/O
<1> ;     THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4 INCH 360 KB,
<1> ;     1.2 MB, 720 KB AND 1.44 MB DISKETTE DRIVES.
<1> ; INPUT
<1> ;     (AH) =  00H RESET DISKETTE SYSTEM
<1> ;            HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED
<1> ;            ON ALL DRIVES
<1> ;----------------------------------------------------------------------------------
<1> ;     (AH)= 01H  READ THE STATUS OF THE SYSTEM INTO (AH)
<1> ;            @DISKETTE_STATUS FROM LAST OPERATION IS USED
<1> ;----------------------------------------------------------------------------------
<1> ;     REGISTERS FOR READ/WRITE/VERIFY/FORMAT
<1> ;     (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
<1> ;     (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
<1> ;     (CH) - TRACK NUMBER (NOT VALUE CHECKED)
<1> ;            MEDIA  DRIVE  TRACK NUMBER
<1> ;            320/360       320/360          0-39
<1> ;            320/360       1.2M      0-39
<1> ;            1.2M   1.2M       0-79
<1> ;            720K   720K       0-79
<1> ;            1.44M  1.44M      0-79
<1> ;     (CL) -       SECTOR NUMBER (NOT VALUE CHECKED, NOT USED FOR FORMAT)
<1> ;            MEDIA  DRIVE  SECTOR NUMBER
<1> ;            320/360       320/360           1-8/9
<1> ;            320/360       1.2M       1-8/9
<1> ;            1.2M   1.2M        1-15
<1> ;            720K   720K        1-9
<1> ;            1.44M  1.44M       1-18
<1> ;     (AL)   NUMBER OF SECTORS (NOT VALUE CHECKED)
<1> ;            MEDIA  DRIVE  MAX NUMBER OF SECTORS
<1> ;            320/360       320/360              8/9
```

```
<1> ;           320/360      1.2M           8/9
<1> ;           1.2M   1.2M         15
<1> ;           720K   720K         9
<1> ;           1.44M  1.44M        18
<1> ;
<1> ;     (ES:BX) - ADDRESS OF BUFFER (NOT REQUIRED FOR VERIFY)
<1> ;
<1> ;-------------------------------------------------------------------------------
<1> ;     (AH)= 02H  READ THE DESIRED SECTORS INTO MEMORY
<1> ;-------------------------------------------------------------------------------
<1> ;     (AH)= 03H  WRITE THE DESIRED SECTORS FROM MEMORY
<1> ;-------------------------------------------------------------------------------
<1> ;     (AH)= 04H  VERIFY THE DESIRED SECTORS
<1> ;-------------------------------------------------------------------------------
<1> ;     (AH)= 05H  FORMAT THE DESIRED TRACK
<1> ;           (ES,BX) MUST POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS
<1> ;           FOR THE      TRACK. EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N),
<1> ;           WHERE C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,
<1> ;           N= NUMBER OF BYTES PER SECTOR (00=128,01=256,02=512,03=1024),
<1> ;           THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
<1> ;           THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
<1> ;           READ/WRITE ACCESS.
<1> ;           PRIOR TO FORMATTING A DISKETTE, IF THERE EXISTS MORE THAN
<1> ;           ONE SUPPORTED MEDIA FORMAT TYPE WITHIN THE DRIVE IN QUESTION,
<1> ;           THEN "SET DASD TYPE" (INT 13H, AH = 17H) OR 'SET MEDIA TYPE'
<1> ;           (INT 13H, AH =  18H) MUST BE CALLED TO SET THE DISKETTE TYPE
<1> ;           THAT IS TO BE FORMATTED. IF "SET DASD TYPE" OR "SET MEDIA TYPE"
<1> ;           IS NOT CALLED, THE FORMAT ROUTINE WILL ASSUME THE
<1> ;           MEDIA FORMAT TO BE THE MAXIMUM CAPACITY OF THE DRIVE.
<1> ;
<1> ;           THESE PARAMETERS OF DISK BASE MUST BE CHANGED IN ORDER TO
<1> ;           FORMAT THE FOLLOWING MEDIAS:
<1> ;           ---------------------------------------------
<1> ;           : MEDIA  :     DRIVE      : PARM 1 : PARM 2 :
<1> ;           ---------------------------------------------
<1> ;           : 320K  : 320K/360K/1.2M :  50H   :   8    :
<1> ;           : 360K  : 320K/360K/1.2M :  50H   :   9    :
<1> ;           : 1.2M  : 1.2M           :  54H   :  15    :
<1> ;           : 720K  : 720K/1.44M     :  50H   :   9    :
<1> ;           : 1.44M       : 1.44M          :  6CH   :  18    :
<1> ;           ---------------------------------------------
<1> ;           NOTES: - PARM 1 = GAP LENGTH FOR FORMAT
<1> ;                  - PARM 2 = EOT (LAST SECTOR ON TRACK)
<1> ;                  - DISK BASE IS POINTED BY DISK POINTER LOCATED
<1> ;                   AT ABSOLUTE ADDRESS 0:78.
<1> ;                  - WHEN FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS
<1> ;                   SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES.
<1> ;-------------------------------------------------------------------------------
<1> ;     (AH) = 08H READ DRIVE PARAMETERS
<1> ;     REGISTERS
<1> ;       INPUT
<1> ;         (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
<1> ;          ** 27/05/2016 - TRDOS 386 (TRDOS v2.0) **
<1> ;            ** EBX = Buffer address for floppy disk parameters table **
<1> ;       OUTPUT
<1> ;         (ES:DI) POINTS TO DRIVE PARAMETER TABLE
<1> ;         *** TRDOS 386 note: floppy disk parameter table (16 bytes)
<1> ;         will be returned to user in EBX, buffer address *** 27/05/2016 ***
<1> ;
<1> ;         (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
<1> ;         (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
<1> ;                BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
<1> ;         (DH) - MAXIMUM HEAD NUMBER
<1> ;         (DL) - NUMBER OF DISKETTE DRIVES INSTALLED
<1> ;         (BH) - 0
<1> ;         (BL) - BITS 7 THRU 4 - 0
<1> ;                BITS 3 THRU 0 - VALID DRIVE TYPE VALUE IN CMOS
<1> ;         (AX) - 0
<1> ;      UNDER THE FOLLOWING CIRCUMSTANCES:
<1> ;         (1) THE DRIVE NUMBER IS INVALID,
<1> ;         (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,
<1> ;         (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,
<1> ;         (4) OR THE DRIVE TYPE IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID
<1> ;         THEN ES,AX,BX,CX,DH,DI=0 ; DL=NUMBER OF DRIVES.
<1> ;         IF NO DRIVES ARE PRESENT THEN: ES,AX,BX,CX,DX,DI=0.
<1> ;         @DISKETTE_STATUS = 0 AND CY IS RESET.
<1> ;-------------------------------------------------------------------------------
<1> ;     (AH)= 15H  READ DASD TYPE
<1> ;     OUTPUT REGISTERS
<1> ;     (AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR
<1> ;           00 - DRIVE NOT PRESENT
<1> ;           01 - DISKETTE, NO CHANGE LINE AVAILABLE
<1> ;           02 - DISKETTE, CHANGE LINE AVAILABLE
<1> ;           03 - RESERVED (FIXED DISK)
<1> ;     (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
<1> ;-------------------------------------------------------------------------------
<1> ;     (AH)= 16H  DISK CHANGE LINE STATUS
<1> ;     OUTPUT REGISTERS
<1> ;     (AH) - 00 - DISK CHANGE LINE NOT ACTIVE
<1> ;            06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
<1> ;     (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
<1> ;-------------------------------------------------------------------------------
<1> ;     (AH)= 17H  SET DASD TYPE FOR FORMAT
<1> ;     INPUT REGISTERS
<1> ;     (AL) - 00 - NOT USED
<1> ;           01 - DISKETTE 320/360K IN 360K DRIVE
<1> ;           02 - DISKETTE 360K IN 1.2M DRIVE
<1> ;           03 - DISKETTE 1.2M IN 1.2M DRIVE
<1> ;           04 - DISKETTE 720K IN 720K DRIVE
<1> ;     (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED:
<1> ;            (DO NOT USE WHEN DISKETTE ATTACH CARD USED)
<1> ;-------------------------------------------------------------------------------
<1> ;     (AH)= 18H  SET MEDIA TYPE FOR FORMAT
```

```
                             <1> ;     INPUT REGISTERS
                             <1> ;     (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM TRACKS
                             <1> ;     (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
                             <1> ;            BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
                             <1> ;     (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHACKED)
                             <1> ;     OUTPUT REGISTERS:
                             <1> ;     (ES:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,
                             <1> ;             UNCHANGED IF (AH) IS NON-ZERO
                             <1> ;     (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
                             <1> ;          - 01H, CY = 1, FUNCTION IS NOT AVAILABLE
                             <1> ;          - 0CH, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED
                             <1> ;          - 80H, CY = 1, TIME OUT (DISKETTE NOT PRESENT)
                             <1> ;-------------------------------------------------------------------------------
                             <1> ;     DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
                             <1> ;     THAN 360 KB DRIVE. IF THE DISK CHANGE LINE IS FOUND TO BE
                             <1> ;     ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
                             <1> ;            ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
                             <1> ;            IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK
                             <1> ;            CHANGE ERROR CODE
                             <1> ;            IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE
                             <1> ;            TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
                             <1> ;     IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.
                             <1> ;
                             <1> ; DATA VARIABLE -- @DISK_POINTER
                             <1> ;     DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; OUTPUT FOR ALL FUNCTIONS
                             <1> ;     AH = STATUS OF OPERATION
                             <1> ;            STATUS BITS ARE DEFINED IN THE EQUATES FOR @DISKETTE_STATUS
                             <1> ;            VARIABLE IN THE DATA SEGMENT OF THIS MODULE
                             <1> ;     CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD
                             <1> ;            TYPE AH=(15)).
                             <1> ;     CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
                             <1> ;     FOR READ/WRITE/VERIFY
                             <1> ;            DS,BX,DX,CX PRESERVED
                             <1> ;     NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
                             <1> ;            ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
                             <1> ;            ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
                             <1> ;            THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
                             <1> ;            PROBLEM IS NOT DUE TO MOTOR START-UP.
                             <1> ;-------------------------------------------------------------------------------
                             <1> ;
                             <1> ; DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
                             <1> ;
                             <1> ;   -----------------------------------------------------------------
                             <1> ;   |       |       |       |       |       |       |       |       |
                             <1> ;   |   7   |   6   |   5   |   4   |   3   |   2   |   1   |   0   |
                             <1> ;   |       |       |       |       |       |       |       |       |
                             <1> ;   -----------------------------------------------------------------
                             <1> ;     |      |      |      |      |      |      |      |
                             <1> ;     |      |      |      |      |      -----------------
                             <1> ;     |      |      |      |      |            |
                             <1> ;     |      |      |      |    RESERVED       |
                             <1> ;     |      |      |      |                PRESENT STATE
                             <1> ;     |      |      |      |     000: 360K IN 360K DRIVE UNESTABLISHED
                             <1> ;     |      |      |      |     001: 360K IN 1.2M DRIVE UNESTABLISHED
                             <1> ;     |      |      |      |     010: 1.2M IN 1.2M DRIVE UNESTABLISHED
                             <1> ;     |      |      |      |     011: 360K IN 360K DRIVE ESTABLISHED
                             <1> ;     |      |      |      |     100: 360K IN 1.2M DRIVE ESTABLISHED
                             <1> ;     |      |      |      |     101: 1.2M IN 1.2M DRIVE ESTABLISHED
                             <1> ;     |      |      |      |     110: RESERVED
                             <1> ;     |      |      |      |     111: NONE OF THE ABOVE
                             <1> ;     |      |      |      |
                             <1> ;     |      |      |      ------>      MEDIA/DRIVE ESTABLISHED
                             <1> ;     |      |      |
                             <1> ;     |      |      -------------->     DOUBLE STEPPING REQUIRED (360K IN 1.2M
                             <1> ;     |      |                     DRIVE)
                             <1> ;     |      |
                             <1> ;     ------------------------------>  DATA TRANSFER RATE FOR THIS DRIVE:
                             <1> ;
                             <1> ;                                      00: 500 KBS
                             <1> ;                                      01: 300 KBS
                             <1> ;                                      10: 250 KBS
                             <1> ;                                      11: RESERVED
                             <1> ;
                             <1> ;
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; STATE OPERATION STARTED - ABSOLUTE ADDRESS 40:92 (DRIVE A) & 93 (DRIVE B)
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40:94 (DRIVE A) & 95 (DRIVE B)
                             <1> ;-------------------------------------------------------------------------------
                             <1>
                             <1> struc MD
00000000 <res 00000001>      <1>       .SPEC1       resb   1      ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00000001 <res 00000001>      <1>       .SPEC2       resb   1      ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00000002 <res 00000001>      <1>       .OFF_TIM     resb   1      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00000003 <res 00000001>      <1>       .BYT_SEC     resb   1      ; 512 BYTES/SECTOR
00000004 <res 00000001>      <1>       .SEC_TRK     resb   1      ; EOT (LAST SECTOR ON TRACK)
00000005 <res 00000001>      <1>       .GAP         resb   1      ; GAP LENGTH
00000006 <res 00000001>      <1>       .DTL         resb   1      ; DTL
00000007 <res 00000001>      <1>       .GAP3        resb   1      ; GAP LENGTH FOR FORMAT
00000008 <res 00000001>      <1>       .FIL_BYT     resb   1      ; FILL BYTE FOR FORMAT
00000009 <res 00000001>      <1>       .HD_TIM            resb   1      ; HEAD SETTLE TIME (MILLISECONDS)
0000000A <res 00000001>      <1>       .STR_TIM     resb   1      ; MOTOR START TIME (1/8 SECONDS)
0000000B <res 00000001>      <1>       .MAX_TRK     resb   1      ; MAX. TRACK NUMBER
0000000C <res 00000001>      <1>       .RATE        resb   1      ; DATA TRANSFER RATE
                             <1> endstruc
                             <1>
                             <1> BIT7OFF     EQU    7FH
                             <1> BIT7ON      EQU    80H
                             <1>
                             <1> ;;int13h: ; 16/02/2015
                             <1> ;; 16/02/2015 - 21/02/2015
                             <1> int40h:
00003373 9C                  <1>       pushfd
00003374 0E                  <1>       push   cs
00003375 E801000000          <1>       call   DISKETTE_IO_1
0000337A C3                  <1>       retn
                             <1> 
                             <1> DISKETTE_IO_1:
                             <1> 
0000337B FB                  <1>       STI                        ; INTERRUPTS BACK ON
0000337C 55                  <1>       PUSH   eBP                 ; USER REGISTER
0000337D 57                  <1>       PUSH   eDI                 ; USER REGISTER
0000337E 52                  <1>       PUSH   eDX                 ; HEAD #, DRIVE # OR USER REGISTER
0000337F 53                  <1>       PUSH   eBX                 ; BUFFER OFFSET PARAMETER OR REGISTER
00003380 51                  <1>       PUSH   eCX                 ; TRACK #-SECTOR # OR USER REGISTER
00003381 89E5                <1>       MOV    eBP,eSP                    ; BP     => PARAMETER LIST DEP. ON AH
                             <1>                                  ; [BP]   = SECTOR #
                             <1>                                  ; [BP+1] = TRACK #
                             <1>                                  ; [BP+2] = BUFFER OFFSET
                             <1>                                  ; FOR RETURN OF DRIVE PARAMETERS:
                             <1>                                  ; CL/[BP] = BITS 7&6 HI BITS OF MAX CYL
                             <1>                                  ;          BITS 0-5 MAX SECTORS/TRACK
                             <1>                                  ; CH/[BP+1] = LOW 8 BITS OF MAX CYL.
                             <1>                                  ; BL/[BP+2] = BITS 7-4 = 0
                             <1>                                  ;            BITS 3-0 = VALID CMOS TYPE
                             <1>                                  ; BH/[BP+3] = 0
                             <1>                                  ; DL/[BP+4] = # DRIVES INSTALLED
                             <1>                                  ; DH/[BP+5] = MAX HEAD #
                             <1>                                  ; DI/[BP+6] = OFFSET TO DISK BASE
00003383 06                  <1>       push   es ; 06/02/2015
00003384 1E                  <1>       PUSH   DS                  ; BUFFER SEGMENT PARM OR USER REGISTER
00003385 56                  <1>       PUSH   eSI                 ; USER REGISTERS
                             <1>       ;CALL  DDS                 ; SEGMENT OF BIOS DATA AREA TO DS
                             <1>       ;mov   cx, cs
                             <1>       ;mov   ds, cx
00003386 66B91000            <1>       mov    cx, KDATA
0000338A 8ED9                <1>         mov     ds, cx
0000338C 8EC1                <1>         mov     es, cx
                             <1> 
                             <1>       ;CMP   AH,(FNC_TAE-FNC_TAB)/2     ; CHECK FOR > LARGEST FUNCTION
0000338E 80FC19              <1>       cmp    ah,(FNC_TAE-FNC_TAB)/4  ; 18/02/2015
00003391 7202                <1>       JB     short OK_FUNC       ; FUNCTION OK
00003393 B414                <1>       MOV    AH,14H              ; REPLACE WITH KNOWN INVALID FUNCTION
                             <1> OK_FUNC:
00003395 80FC01              <1>       CMP    AH,1                ; RESET OR STATUS ?
00003398 760C                <1>       JBE    short OK_DRV        ; IF RESET OR STATUS DRIVE ALWAYS OK
0000339A 80FC08              <1>       CMP    AH,8                ; READ DRIVE PARMS ?
0000339D 7407                <1>       JZ     short OK_DRV        ; IF SO DRIVE CHECKED LATER
0000339F 80FA01              <1>       CMP    DL,1                ; DRIVES 0 AND 1 OK
000033A2 7602                <1>       JBE    short OK_DRV        ; IF 0 OR 1 THEN JUMP
000033A4 B414                <1>       MOV    AH,14H              ; REPLACE WITH KNOWN INVALID FUNCTION
                             <1> OK_DRV:
000033A6 31C9                <1>       xor    ecx, ecx
                             <1>       ;mov   esi, ecx ; 08/02/2015
000033A8 89CF                <1>       mov    edi, ecx ; 08/02/2015
000033AA 88E1                <1>       MOV    CL,AH               ; CL = FUNCTION
                             <1>       ;XOR   CH,CH               ; CX = FUNCTION
                             <1>       ;SHL   CL, 1               ; FUNCTION TIMES 2
000033AC C0E102              <1>       SHL    CL, 2 ; 20/02/2015  ; FUNCTION TIMES 4 (for 32 bit offset)
000033AF BB[E7330000]        <1>       MOV    eBX,FNC_TAB         ; LOAD START OF FUNCTION TABLE
000033B4 01CB                <1>       ADD    eBX,eCX                    ; ADD OFFSET INTO TABLE => ROUTINE
000033B6 88F4                <1>       MOV    AH,DH               ; AX = HEAD #,# OF SECTORS OR DASD TYPE
000033B8 30F6                <1>       XOR    DH,DH               ; DX = DRIVE #
000033BA 6689C6              <1>       MOV    SI,AX               ; SI = HEAD #,# OF SECTORS OR DASD TYPE
000033BD 6689D7              <1>       MOV      DI,DX                    ; DI = DRIVE #
                             <1>       ;
                             <1>       ; 11/12/2014
000033C0 8815[E55C0000]      <1>         mov     [cfd], dl               ; current floppy drive (for 'GET_PARM')
                             <1>       ;
000033C6 8A25[C0580100]      <1>       MOV    AH, [DSKETTE_STATUS]      ; LOAD STATUS TO AH FOR STATUS FUNCTION
000033CC C605[C0580100]00    <1>       MOV    byte [DSKETTE_STATUS],0   ; INITIALIZE FOR ALL OTHERS
                             <1> 
                             <1> ;     THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
                             <1> ;     THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
                             <1> ;     FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.
                             <1> ;
                             <1> ;            DI     : DRIVE #
                             <1> ;            SI-HI  : HEAD #
                             <1> ;            SI-LOW : # OF SECTORS OR DASD TYPE FOR FORMAT
                             <1> ;            ES     : BUFFER SEGMENT
                             <1> ;            [BP]   : SECTOR #
                             <1> ;            [BP+1] : TRACK #
                             <1> ;            [BP+2] : BUFFER OFFSET
                             <1> ;
                             <1> ;     ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
                             <1> ;     SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
                             <1> ;     CONDITION). IN MOST CASES, WHEN CY = 1, @DSKETTE_STATUS CONTAINS THE
                             <1> ;     SPECIFIC ERROR CODE.
                             <1> ;
                             <1>                                  ; (AH) = @DSKETTE_STATUS
000033D3 FF13                <1>       CALL   dWORD [eBX]         ; CALL THE REQUESTED FUNCTION
000033D5 5E                  <1>       POP    eSI                 ; RESTORE ALL REGISTERS
000033D6 1F                  <1>       POP    DS
000033D7 07                  <1>       pop    es     ; 06/02/2015
000033D8 59                  <1>       POP    eCX
000033D9 5B                  <1>       POP    eBX
000033DA 5A                  <1>       POP    eDX
000033DB 5F                  <1>       POP    eDI
000033DC 89E5                <1>       MOV    eBP, eSP
000033DE 50                  <1>       PUSH   eAX
000033DF 9C                  <1>       PUSHFd
000033E0 58                  <1>       POP    eAX
                             <1>       ;MOV   [BP+6], AX
000033E1 89450C              <1>       mov    [ebp+12], eax  ; 18/02/2015, flags
```

```
000033E4 58                     <1>       POP    eAX
000033E5 5D                     <1>       POP    eBP
000033E6 CF                     <1>       IRETd
                                <1>
                                <1> ;----------------------------------------------------------------------------
                                <1> ; DW --> dd (06/02/2015)
000033E7 [4B340000]             <1> FNC_TAB      dd     DSK_RESET           ; AH = 00H; RESET
000033EB [C4340000]             <1>       dd     DSK_STATUS          ; AH = 01H; STATUS
000033EF [D5340000]             <1>       dd     DSK_READ            ; AH = 02H; READ
000033F3 [E6340000]             <1>       dd     DSK_WRITE           ; AH = 03H; WRITE
000033F7 [F7340000]             <1>       dd     DSK_VERF            ; AH = 04H; VERIFY
000033FB [08350000]             <1>       dd     DSK_FORMAT          ; AH = 05H; FORMAT
000033FF [8D350000]             <1>       dd     FNC_ERR                    ; AH = 06H; INVALID
00003403 [8D350000]             <1>       dd     FNC_ERR                    ; AH = 07H; INVALID
00003407 [9A350000]             <1>       dd     DSK_PARMS           ; AH = 08H; READ DRIVE PARAMETERS
0000340B [8D350000]             <1>       dd     FNC_ERR                    ; AH = 09H; INVALID
0000340F [8D350000]             <1>       dd     FNC_ERR                    ; AH = 0AH; INVALID
00003413 [8D350000]             <1>       dd     FNC_ERR                    ; AH = 0BH; INVALID
00003417 [8D350000]             <1>       dd     FNC_ERR                    ; AH = 0CH; INVALID
0000341B [8D350000]             <1>       dd     FNC_ERR                    ; AH = 0DH; INVALID
0000341F [8D350000]             <1>       dd     FNC_ERR                    ; AH = 0EH; INVALID
00003423 [8D350000]             <1>       dd     FNC_ERR                    ; AH = 0FH; INVALID
00003427 [8D350000]             <1>       dd     FNC_ERR                    ; AH = 10H; INVALID
0000342B [8D350000]             <1>       dd     FNC_ERR                    ; AH = 11H; INVALID
0000342F [8D350000]             <1>       dd     FNC_ERR                    ; AH = 12H; INVALID
00003433 [8D350000]             <1>       dd     FNC_ERR                    ; AH = 13H; INVALID
00003437 [8D350000]             <1>       dd     FNC_ERR                    ; AH = 14H; INVALID
0000343B [72360000]             <1>       dd     DSK_TYPE            ; AH = 15H; READ DASD TYPE
0000343F [9D360000]             <1>       dd     DSK_CHANGE          ; AH = 16H; CHANGE STATUS
00003443 [D7360000]             <1>       dd     FORMAT_SET          ; AH = 17H; SET DASD TYPE
00003447 [5A370000]             <1>       dd     SET_MEDIA           ; AH = 18H; SET MEDIA TYPE
                                <1> FNC_TAE EQU     $                         ; END
                                <1>
                                <1> ;----------------------------------------------------------------------------
                                <1> ; DISK_RESET (AH = 00H)
                                <1> ;           RESET THE DISKETTE SYSTEM.
                                <1> ;
                                <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                                <1> ;----------------------------------------------------------------------------
                                <1> DSK_RESET:
0000344B 66BAF203               <1>       MOV    DX,03F2H            ; ADAPTER CONTROL PORT
0000344F FA                     <1>       CLI                        ; NO INTERRUPTS
00003450 A0[BE580100]           <1>       MOV    AL,[MOTOR_STATUS]   ; GET DIGITAL OUTPUT REGISTER REFLECTION
00003455 243F                   <1>       AND    AL,00111111B        ; KEEP SELECTED AND MOTOR ON BITS
00003457 C0C004                 <1>       ROL    AL,4                ; MOTOR VALUE TO HIGH NIBBLE
                                <1>                                  ; DRIVE SELECT TO LOW NIBBLE
0000345A 0C08                   <1>       OR     AL,00001000B        ; TURN ON INTERRUPT ENABLE
0000345C EE                     <1>       OUT    DX,AL               ; RESET THE ADAPTER
0000345D C605[BD580100]00       <1>       MOV    byte [SEEK_STATUS],0       ; SET RECALIBRATE REQUIRED ON ALL DRIVES
                                <1>       ;JMP   $+2                 ; WAIT FOR I/O
                                <1>       ;JMP   $+2                 ; WAIT FOR I/O (TO INSURE MINIMUM
                                <1>                                  ;      PULSE WIDTH)
                                <1>       ; 19/12/2014
                                <1>       NEWIODELAY
00003464 E6EB                   <2>  out 0ebh,al
                                <1>
                                <1>       ; 17/12/2014
                                <1>       ; AWARD BIOS 1999 - RESETDRIVES (ADISK.ASM)
00003466 B915000000             <1>       mov    ecx, WAITCPU_RESET_ON     ; cx = 21 -- Min. 14 micro seconds !?
                                <1> wdw1:
                                <1>       NEWIODELAY   ; 27/02/2015
0000346B E6EB                   <2>  out 0ebh,al
0000346D E2FC                   <1>       loop   wdw1
                                <1>       ;
0000346F 0C04                   <1>       OR     AL,00000100B        ; TURN OFF RESET BIT
00003471 EE                     <1>       OUT    DX,AL               ; RESET THE ADAPTER
                                <1>       ; 16/12/2014
                                <1>       IODELAY
00003472 EB00                   <2>  jmp short $+2
00003474 EB00                   <2>  jmp short $+2
                                <1>       ;
                                <1>       ;STI                       ; ENABLE THE INTERRUPTS
00003476 E83C0C0000             <1>       CALL   WAIT_INT            ; WAIT FOR THE INTERRUPT
0000347B 723E                   <1>       JC     short DR_ERR        ; IF ERROR, RETURN IT
0000347D 66B9C000               <1>       MOV    CX,11000000B        ; CL = EXPECTED @NEC_STATUS
                                <1> NXT_DRV:
00003481 6651                   <1>       PUSH   CX                  ; SAVE FOR CALL
00003483 B8[B9340000]           <1>       MOV    eAX, DR_POP_ERR     ; LOAD NEC_OUTPUT ERROR ADDRESS
00003488 50                     <1>       PUSH   eAX                 ; "
00003489 B408                   <1>       MOV    AH,08H              ; SENSE INTERRUPT STATUS COMMAND
0000348B E81A0B0000             <1>       CALL   NEC_OUTPUT
00003490 58                     <1>       POP    eAX                 ; THROW AWAY ERROR RETURN
00003491 E8510C0000             <1>       CALL   RESULTS                    ; READ IN THE RESULTS
00003496 6659                   <1>       POP    CX                  ; RESTORE AFTER CALL
00003498 7221                   <1>       JC     short DR_ERR        ; ERROR RETURN
0000349A 3A0D[C1580100]         <1>       CMP    CL, [NEC_STATUS]    ; TEST FOR DRIVE READY TRANSITION
000034A0 7519                   <1>       JNZ    short DR_ERR        ; EVERYTHING OK
000034A2 FEC1                   <1>       INC    CL                  ; NEXT EXPECTED @NEC_STATUS
000034A4 80F9C3                 <1>       CMP    CL,11000011B        ; ALL POSSIBLE DRIVES CLEARED
000034A7 76D8                   <1>       JBE    short NXT_DRV       ; FALL THRU IF 11000100B OR >
                                <1>       ;
000034A9 E869030000             <1>       CALL   SEND_SPEC           ; SEND SPECIFY COMMAND TO NEC
                                <1> RESBAC:
000034AE E81D090000             <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
000034B3 6689F3                 <1>       MOV    BX,SI               ; GET SAVED AL TO BL
000034B6 88D8                   <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
000034B8 C3                     <1>       RETn
                                <1> DR_POP_ERR:
000034B9 6659                   <1>       POP    CX                  ; CLEAR STACK
                                <1> DR_ERR:
000034BB 800D[C0580100]20       <1>       OR     byte [DSKETTE_STATUS],BAD_NEC ; SET ERROR CODE
000034C2 EBEA                   <1>       JMP    SHORT RESBAC        ; RETURN FROM RESET
                                <1>
```

```
                            <1> ;-------------------------------------------------------------------------------
                            <1> ; DISK_STATUS      (AH = 01H)
                            <1> ;     DISKETTE STATUS.
                            <1> ;
                            <1> ; ON ENTRY:  AH : STATUS OF PREVIOUS OPERATION
                            <1> ;
                            <1> ; ON EXIT:   AH, @DSKETTE_STATUS, CY REFLECT STATUS OF PREVIOUS OPERATION.
                            <1> ;-------------------------------------------------------------------------------
                            <1> DSK_STATUS:
000034C4 8825[C0580100]     <1>       MOV    [DSKETTE_STATUS],AH ; PUT BACK FOR SETUP END
000034CA E801090000         <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
000034CF 6689F3             <1>       MOV    BX,SI               ; GET SAVED AL TO BL
000034D2 88D8               <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
000034D4 C3                 <1>       RETn
                            <1>
                            <1> ;-------------------------------------------------------------------------------
                            <1> ; DISK_READ  (AH = 02H)
                            <1> ;     DISKETTE READ.
                            <1> ;
                            <1> ; ON ENTRY:  DI     : DRIVE #
                            <1> ;            SI-HI  : HEAD #
                            <1> ;            SI-LOW : # OF SECTORS
                            <1> ;            ES     : BUFFER SEGMENT
                            <1> ;            [BP]   : SECTOR #
                            <1> ;            [BP+1] : TRACK #
                            <1> ;            [BP+2] : BUFFER OFFSET
                            <1> ;
                            <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                            <1> ;-------------------------------------------------------------------------------
                            <1>
                            <1> ; 06/02/2015, ES:BX -> EBX (unix386.s)
                            <1>
                            <1> DSK_READ:
000034D5 8025[BE580100]7F   <1>       AND    byte [MOTOR_STATUS],01111111B ; INDICATE A READ OPERATION
000034DC 66B846E6           <1>       MOV    AX,0E646H           ; AX = NEC COMMAND, DMA COMMAND
000034E0 E83C040000         <1>       CALL   RD_WR_VF            ; COMMON READ/WRITE/VERIFY
000034E5 C3                 <1>       RETn
                            <1>
                            <1> ;-------------------------------------------------------------------------------
                            <1> ; DISK_WRITE (AH = 03H)
                            <1> ;     DISKETTE WRITE.
                            <1> ;
                            <1> ; ON ENTRY:  DI     : DRIVE #
                            <1> ;            SI-HI  : HEAD #
                            <1> ;            SI-LOW : # OF SECTORS
                            <1> ;            ES     : BUFFER SEGMENT
                            <1> ;            [BP]   : SECTOR #
                            <1> ;            [BP+1] : TRACK #
                            <1> ;            [BP+2] : BUFFER OFFSET
                            <1> ;
                            <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                            <1> ;-------------------------------------------------------------------------------
                            <1>
                            <1> ; 06/02/2015, ES:BX -> EBX (unix386.s)
                            <1>
                            <1> DSK_WRITE:
000034E6 66B84AC5           <1>       MOV    AX,0C54AH           ; AX = NEC COMMAND, DMA COMMAND
000034EA 800D[BE580100]80   <1>         OR      byte [MOTOR_STATUS],10000000B ; INDICATE WRITE OPERATION
000034F1 E82B040000         <1>       CALL   RD_WR_VF            ; COMMON READ/WRITE/VERIFY
000034F6 C3                 <1>       RETn
                            <1>
                            <1> ;-------------------------------------------------------------------------------
                            <1> ; DISK_VERF  (AH = 04H)
                            <1> ;     DISKETTE VERIFY.
                            <1> ;
                            <1> ; ON ENTRY:  DI     : DRIVE #
                            <1> ;            SI-HI  : HEAD #
                            <1> ;            SI-LOW : # OF SECTORS
                            <1> ;            ES     : BUFFER SEGMENT
                            <1> ;            [BP]   : SECTOR #
                            <1> ;            [BP+1] : TRACK #
                            <1> ;            [BP+2] : BUFFER OFFSET
                            <1> ;
                            <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                            <1> ;-------------------------------------------------------------------------------
                            <1> DSK_VERF:
000034F7 8025[BE580100]7F   <1>       AND    byte [MOTOR_STATUS],01111111B ; INDICATE A READ OPERATION
000034FE 66B842E6           <1>       MOV    AX,0E642H           ; AX = NEC COMMAND, DMA COMMAND
00003502 E81A040000         <1>       CALL   RD_WR_VF            ; COMMON READ/WRITE/VERIFY
00003507 C3                 <1>       RETn
                            <1>
                            <1> ;-------------------------------------------------------------------------------
                            <1> ; DISK_FORMAT      (AH = 05H)
                            <1> ;     DISKETTE FORMAT.
                            <1> ;
                            <1> ; ON ENTRY:  DI     : DRIVE #
                            <1> ;            SI-HI  : HEAD #
                            <1> ;            SI-LOW : # OF SECTORS
                            <1> ;            ES     : BUFFER SEGMENT
                            <1> ;            [BP]   : SECTOR #
                            <1> ;            [BP+1] : TRACK #
                            <1> ;            [BP+2] : BUFFER OFFSET
                            <1> ;            @DISK_POINTER POINTS TO THE PARAMETER TABLE OF THIS DRIVE
                            <1> ;
                            <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                            <1> ;-------------------------------------------------------------------------------
                            <1> DSK_FORMAT:
00003508 E853030000         <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
0000350D E84F050000         <1>       CALL   FMT_INIT            ; ESTABLISH STATE IF UNESTABLISHED
00003512 800D[BE580100]80   <1>         OR      byte [MOTOR_STATUS], 10000000B ; INDICATE WRITE OPERATION
00003519 E897050000         <1>       CALL   MED_CHANGE          ; CHECK MEDIA CHANGE AND RESET IF SO
0000351E 725D               <1>         JC      short FM_DON           ; MEDIA CHANGED, SKIP
00003520 E8F2020000         <1>       CALL   SEND_SPEC           ; SEND SPECIFY COMMAND TO NEC
```

```
00003525 E8FD050000          <1>       CALL   CHK_LASTRATE        ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
0000352A 7405                <1>         JZ      short FM_WR             ; YES, SKIP SPECIFY COMMAND
0000352C E8D4050000          <1>       CALL   SEND_RATE           ; SEND DATA RATE TO CONTROLLER
                             <1> FM_WR:
00003531 E88A060000          <1>       CALL   FMTDMA_SET          ; SET UP THE DMA FOR FORMAT
00003536 7245                <1>         JC      short FM_DON            ; RETURN WITH ERROR
00003538 B44D                <1>       MOV    AH,04DH                  ; ESTABLISH THE FORMAT COMMAND
0000353A E8E7060000          <1>       CALL   NEC_INIT            ; INITIALIZE THE NEC
0000353F 723C                <1>         JC      short FM_DON            ; ERROR - EXIT
00003541 B8[7D350000]        <1>         MOV     eAX, FM_DON             ; LOAD ERROR ADDRESS
00003546 50                  <1>       PUSH   eAX                 ; PUSH NEC_OUT ERROR RETURN
00003547 B203                <1>       MOV    DL,3                ; BYTES/SECTOR VALUE TO NEC
00003549 E856090000          <1>       CALL   GET_PARM
0000354E E8570A0000          <1>       CALL   NEC_OUTPUT
00003553 B204                <1>       MOV    DL,4                ; SECTORS/TRACK VALUE TO NEC
00003555 E84A090000          <1>       CALL   GET_PARM
0000355A E84B0A0000          <1>       CALL   NEC_OUTPUT
0000355F B207                <1>       MOV    DL,7                ; GAP LENGTH VALUE TO NEC
00003561 E83E090000          <1>       CALL   GET_PARM
00003566 E83F0A0000          <1>       CALL   NEC_OUTPUT
0000356B B208                <1>       MOV    DL,8                ; FILLER BYTE TO NEC
0000356D E832090000          <1>       CALL   GET_PARM
00003572 E8330A0000          <1>       CALL   NEC_OUTPUT
00003577 58                  <1>       POP    eAX                 ; THROW AWAY ERROR
00003578 E827070000          <1>       CALL   NEC_TERM            ; TERMINATE, RECEIVE STATUS, ETC,
                             <1> FM_DON:
0000357D E80F030000          <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
00003582 E849080000          <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
00003587 6689F3              <1>       MOV    BX,SI               ; GET SAVED AL TO BL
0000358A 88D8                <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
0000358C C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; FNC_ERR
                             <1> ;     INVALID FUNCTION REQUESTED OR INVALID DRIVE:
                             <1> ;     SET BAD COMMAND IN STATUS.
                             <1> ;
                             <1> ; ON EXIT:  @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;-------------------------------------------------------------------------------
                             <1> FNC_ERR:                         ; INVALID FUNCTION REQUEST
0000358D 6689F0              <1>       MOV    AX,SI               ; RESTORE AL
00003590 B401                <1>       MOV    AH,BAD_CMD          ; SET BAD COMMAND ERROR
00003592 8825[C0580100]      <1>       MOV    [DSKETTE_STATUS],AH ; STORE IN DATA AREA
00003598 F9                  <1>       STC                        ; SET CARRY INDICATING ERROR
00003599 C3                  <1>       RETn
                             <1>
                             <1> ; 01/06/2016
                             <1> ; 28/05/2016
                             <1> ; 27/05/2016 - TRDOS 386 (TRDOS v.2.0)
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; DISK_PARMS (AH = 08H)
                             <1> ;     READ DRIVE PARAMETERS.
                             <1> ;
                             <1> ; ON ENTRY:  DI : DRIVE #
                             <1> ;            ; 27/05/2016
                             <1> ;            EBX = Buffer Address for floppy disk parameters table (16 bytes)
                             <1> ;
                             <1> ; ON EXIT:   CL/[BP]   = BITS 7 & 6 HI 2 BITS OF MAX CYLINDER
                             <1> ;                        BITS 0-5 MAX SECTORS/TRACK
                             <1> ;            CH/[BP+1] = LOW 8 BITS OF MAX CYLINDER
                             <1> ;            BL/[BP+2] = BITS 7-4 = 0
                             <1> ;                        BITS 3-0 = VALID CMOS DRIVE TYPE
                             <1> ;            BH/[BP+3] = 0
                             <1> ;            DL/[BP+4] = # DRIVES INSTALLED (VALUE CHECKED)
                             <1> ;            DH/[BP+5] = MAX HEAD #
                             <1> ;          ** 27/05/2016 - TRDOS 386 (TRDOS v2.0) **
                             <1> ;            ** EBX = Buffer address for floppy disk parameters table **
                             <1> ;            ;DI/[BP+6] = OFFSET TO DISK_BASE
                             <1> ;            ;ES        = SEGMENT OF DISK_BASE
                             <1> ;
                             <1> ;            AX        = 0
                             <1> ;
                             <1> ;            NOTE : THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT
                             <1> ;                   THE LOCATIONS WHERE THE MAIN ROUTINE WILL POP THEM
                             <1> ;                   INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE
                             <1> ;                   CALLER.
                             <1> ;-------------------------------------------------------------------------------
                             <1> DSK_PARMS:
0000359A E8C1020000          <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH,
                             <1>      ;      MOV    WORD [BP+2],0       ; DRIVE TYPE = 0
                             <1>      ;  MOV     AX, [EQUIP_FLAG]        ; LOAD EQUIPMENT FLAG FOR # DISKETTES
                             <1>      ;  AND     AL,11000001B            ; KEEP DISKETTE DRIVE BITS
                             <1>      ;  MOV     DL,2                    ; DISKETTE DRIVES = 2
                             <1>      ;  CMP     AL,01000001B            ; 2 DRIVES INSTALLED ?
                             <1>      ;  JZ      short STO_DL            ; IF YES JUMP
                             <1>      ;  DEC     DL                      ; DISKETTE DRIVES = 1
                             <1>      ;  CMP     AL,00000001B            ; 1 DRIVE INSTALLED ?
                             <1>      ;  JNZ     short NON_DRV           ; IF NO JUMP
0000359F 29D2                <1>       sub    edx, edx
000035A1 66A1[F65C0000]      <1>       mov     ax, [fd0_type]
000035A7 6621C0              <1>       and     ax, ax
000035AA 0F848A000000        <1>         jz      NON_DRV
000035B0 FEC2                <1>       inc     dl
000035B2 20E4                <1>       and     ah, ah
000035B4 7402                <1>       jz     short STO_DL
000035B6 FEC2                <1>       inc     dl
                             <1> STO_DL:
                             <1>       ;MOV   [BP+4],DL           ; STORE NUMBER OF DRIVES
000035B8 895508              <1>       mov    [ebp+8], edx ; 20/02/2015
000035BB 6683FF01            <1>       CMP    DI,1                ; CHECK FOR VALID DRIVE
000035BF 777C                <1>       JA     short NON_DRV1            ; DRIVE INVALID
                             <1>       ;MOV   BYTE [BP+5],1       ; MAXIMUM HEAD NUMBER =    1
000035C1 C6450901            <1>       mov    byte [ebp+9], 1  ; 20/02/2015
```

```
000035C5 E8D1080000           <1>       CALL   CMOS_TYPE           ; RETURN DRIVE TYPE IN AL
                              <1>       ;;20/02/2015
                              <1>       ;;JC   short CHK_EST       ; IF CMOS BAD CHECKSUM ESTABLISHED
                              <1>       ;;OR   AL,AL               ; TEST FOR NO DRIVE TYPE
000035CA 740F                 <1>       JZ     short CHK_EST       ; JUMP IF SO
000035CC E81B020000           <1>       CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
000035D1 7208                 <1>       JC     short CHK_EST       ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
                              <1>       ;MOV   [BP+2],AL           ; STORE VALID CMOS DRIVE TYPE
                              <1>         ;mov [ebp+4], al ; 06/02/2015
000035D3 8A4B04               <1>       MOV     CL, [eBX+MD.SEC_TRK]     ; GET SECTOR/TRACK
000035D6 8A6B0B               <1>         MOV     CH, [eBX+MD.MAX_TRK]     ; GET MAX. TRACK NUMBER
000035D9 EB36                 <1>       JMP    SHORT STO_CX        ; CMOS GOOD, USE CMOS
                              <1> CHK_EST:
000035DB 8AA7[CD580100]       <1>       MOV    AH, [DSK_STATE+eDI] ; LOAD STATE FOR THIS DRIVE
000035E1 F6C410               <1>       TEST   AH,MED_DET          ; CHECK FOR ESTABLISHED STATE
000035E4 7457                 <1>       JZ     short NON_DRV1             ; CMOS BAD/INVALID OR UNESTABLISHED
                              <1> USE_EST:
000035E6 80E4C0               <1>       AND    AH,RATE_MSK         ; ISOLATE STATE
000035E9 80FC80               <1>       CMP    AH,RATE_250         ; RATE 250 ?
000035EC 7570                 <1>       JNE    short USE_EST2             ; NO, GO CHECK OTHER RATE
                              <1>
                              <1> ;-----      DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST
                              <1>
000035EE B001                 <1>       MOV    AL,01               ; DRIVE TYPE 1 (360KB)
000035F0 E8F7010000           <1>       CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
000035F5 8A4B04               <1>         MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
000035F8 8A6B0B               <1>         MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
000035FB F687[CD580100]01     <1>       TEST   byte [DSK_STATE+eDI],TRK_CAPA ; 80 TRACK ?
00003602 740D                 <1>       JZ     short STO_CX        ; MUST BE 360KB DRIVE
                              <1>
                              <1> ;-----      IT IS 1.44 MB DRIVE
                              <1>
                              <1> PARM144:
00003604 B004                 <1>       MOV    AL,04               ; DRIVE TYPE 4 (1.44MB)
00003606 E8E1010000           <1>       CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
0000360B 8A4B04               <1>         MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
0000360E 8A6B0B               <1>         MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
                              <1> STO_CX:
00003611 894D00               <1>       MOV    [eBP],eCX           ; SAVE POINTER IN STACK FOR RETURN
                              <1> ES_DI:
                              <1>       ;MOV   [BP+6],BX           ; ADDRESS OF MEDIA/DRIVE PARM TABLE
                              <1>       ;mov   [ebp+12], ebx ; 06/02/2015
                              <1>       ;MOV   AX,CS               ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
                              <1>       ;MOV   ES,AX               ; ES IS SEGMENT OF TABLE
                              <1>       ;
                              <1>       ; 28/05/2016
                              <1>       ; 27/05/2016
                              <1>       ; return floppy disk parameters table to user
                              <1>       ; in user's buffer, which is pointed by EBX
                              <1>       ;
00003614 57                   <1>       push   edi
00003615 8B7D04               <1>       mov    edi, [ebp+4]               ; ebx (input), user's buffer address
00003618 0FB6C0               <1>       movzx  eax, al
0000361B 894504               <1>         mov  [ebp+4], eax   ; ebx       ; drive type (for floppy drives)
                              <1>       ; 01/06/2016 (INT 33h, disk type return for floppy disks, in BL)
0000361E A3[C8650100]         <1>       mov    [user_buffer], eax  ; 01/06/2016 (overwrite ebx return value)
                              <1>       ;(INT 33h, Function 08h will replace user's buffer addr with disk type!)
                              <1>       ;
00003623 89DE                 <1>       mov    esi, ebx            ; floppy disk parameter table (16 bytes)
00003625 B910000000           <1>       mov    ecx, 16 ; 16 bytes
0000362A E84AB10000           <1>         call    transfer_to_user_buffer ; trdosk6.s (16/05/2016)
0000362F 5F                   <1>       pop    edi
                              <1> DP_OUT:
00003630 E85C020000           <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
00003635 6631C0               <1>       XOR    AX,AX               ; CLEAR
00003638 F8                   <1>       CLC
00003639 C3                   <1>       RETn
                              <1>
                              <1> ;-----      NO DRIYE PRESENT HANDLER
                              <1>
                              <1> NON_DRV:
                              <1>       ;MOV   BYTE [BP+4],0       ; CLEAR NUMBER OF DRIVES
0000363A 895508               <1>       mov    [ebp+8], edx ; 0 ; 20/02/2015
                              <1> NON_DRV1:
0000363D 6681FF8000           <1>       CMP    DI,80H              ; CHECK FOR FIXED MEDIA TYPE REQUEST
00003642 720C                 <1>       JB     short NON_DRV2             ; CONTINUE IF NOT REQUEST FALL THROUGH
                              <1>
                              <1> ;-----      FIXED DISK REQUEST FALL THROUGH ERROR
                              <1>
00003644 E848020000           <1>       CALL   XLAT_OLD            ; ELSE TRANSLATE TO COMPATIBLE MODE
00003649 6689F0               <1>       MOV    AX,SI               ; RESTORE AL
0000364C B401                 <1>       MOV    AH,BAD_CMD          ; SET BAD COMMAND ERROR
0000364E F9                   <1>       STC
0000364F C3                   <1>       RETn
                              <1>
                              <1> NON_DRV2:
                              <1>       ;XOR   AX,AX               ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
00003650 31C0                 <1>       xor    eax, eax
00003652 66894500             <1>       MOV    [eBP],AX            ; TRACKS, SECTORS/TRACK = 0
                              <1>       ;MOV   [BP+5],AH           ; HEAD = 0
00003656 886509               <1>       mov    [ebp+9], ah ; 06/02/2015
                              <1>       ;MOV   [BP+6],AX           ; OFFSET TO DISK_BASE = 0
00003659 89450C               <1>       mov    [ebp+12], eax
                              <1>       ;MOV   ES,AX               ; ES IS SEGMENT OF TABLE
0000365C EBD2                 <1>       JMP    SHORT DP_OUT
                              <1>
                              <1> ;-----      DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST
                              <1>
                              <1> USE_EST2:
0000365E B002                 <1>       MOV    AL,02               ; DRIVE TYPE 2 (1.2MB)
00003660 E887010000           <1>       CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
00003665 8A4B04               <1>         MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
00003668 8A6B0B               <1>         MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
```

```
0000366B 80FC40              <1>       CMP   AH,RATE_300         ; RATE 300 ?
0000366E 74A1                <1>       JZ    short STO_CX        ; MUST BE 1.2MB DRIVE
00003670 EB92                <1>       JMP   SHORT PARM144       ; ELSE, IT IS 1.44MB DRIVE
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; DISK_TYPE (AH = 15H)
                             <1> ;     THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.
                             <1> ;
                             <1> ;  ON ENTRY: DI = DRIVE #
                             <1> ;
                             <1> ;  ON EXIT:  AH = DRIVE TYPE, CY=0
                             <1> ;-------------------------------------------------------------------------------
                             <1> DSK_TYPE:
00003672 E8E9010000          <1>       CALL  XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
00003677 8A87[CD580100]      <1>       MOV   AL, [DSK_STATE+eDI] ; GET PRESENT STATE INFORMATION
0000367D 08C0                <1>       OR    AL,AL               ; CHECK FOR NO DRIVE
0000367F 7418                <1>       JZ    short NO_DRV
00003681 B401                <1>       MOV   AH,NOCHGLN          ; NO CHANGE LINE FOR 40 TRACK DRIVE
00003683 A801                <1>       TEST  AL,TRK_CAPA         ; IS THIS DRIVE AN 80 TRACK DRIVE?
00003685 7402                <1>       JZ    short DT_BACK              ; IF NO JUMP
00003687 B402                <1>       MOV   AH,CHGLN            ; CHANGE LINE FOR 80 TRACK DRIVE
                             <1> DT_BACK:
00003689 6650                <1>       PUSH  AX                  ; SAVE RETURN VALUE
0000368B E801020000          <1>       CALL  XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
00003690 6658                <1>       POP   AX                  ; RESTORE RETURN VALUE
00003692 F8                  <1>       CLC                       ; NO ERROR
00003693 6689F3              <1>       MOV   BX,SI               ; GET SAVED AL TO BL
00003696 88D8                <1>       MOV   AL,BL               ; PUT BACK FOR RETURN
00003698 C3                  <1>       RETn
                             <1> NO_DRV:
00003699 30E4                <1>       XOR   AH,AH               ; NO DRIVE PRESENT OR UNKNOWN
0000369B EBEC                <1>       JMP   SHORT DT_BACK
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; DISK_CHANGE      (AH = 16H)
                             <1> ;     THIS ROUTINE RETURNS THE STATE OF THE DISK CHANGE LINE.
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ;
                             <1> ; ON EXIT:   AH = @DSKETTE_STATUS
                             <1> ;                00 - DISK CHANGE LINE INACTIVE, CY = 0
                             <1> ;                06 - DISK CHANGE LINE ACTIVE, CY = 1
                             <1> ;-------------------------------------------------------------------------------
                             <1> DSK_CHANGE:
0000369D E8BE010000          <1>       CALL  XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
000036A2 8A87[CD580100]      <1>       MOV   AL, [DSK_STATE+eDI] ; GET MEDIA STATE INFORMATION
000036A8 08C0                <1>       OR    AL,AL               ; DRIVE PRESENT ?
000036AA 7422                <1>       JZ    short DC_NON        ; JUMP IF NO DRIVE
000036AC A801                <1>       TEST  AL,TRK_CAPA         ; 80 TRACK DRIVE ?
000036AE 7407                <1>       JZ    short SETIT         ; IF SO , CHECK CHANGE LINE
                             <1> DC0:
000036B0 E88D0A0000          <1>         CALL    READ_DSKCHNG            ; GO CHECK STATE OF DISK CHANGE LINE
000036B5 7407                <1>       JZ    short FINIS         ; CHANGE LINE NOT ACTIVE
                             <1>
000036B7 C605[C0580100]06    <1> SETIT:      MOV    byte [DSKETTE_STATUS], MEDIA_CHANGE ; INDICATE MEDIA REMOVED
                             <1>
000036BE E8CE010000          <1> FINIS:      CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
000036C3 E808070000          <1>       CALL  SETUP_END           ; VARIOUS CLEANUPS
000036C8 6689F3              <1>       MOV   BX,SI               ; GET SAVED AL TO BL
000036CB 88D8                <1>       MOV   AL,BL               ; PUT BACK FOR RETURN
000036CD C3                  <1>       RETn
                             <1> DC_NON:
000036CE 800D[C0580100]80    <1>       OR    byte [DSKETTE_STATUS], TIME_OUT ; SET TIMEOUT, NO DRIVE
000036D5 EBE7                <1>       JMP   SHORT FINIS
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; FORMAT_SET (AH = 17H)
                             <1> ;     THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF MEDIA TO BE USED
                             <1> ;     FOR THE FOLLOWING FORMAT OPERATION.
                             <1> ;
                             <1> ; ON ENTRY:  SI LOW = DASD TYPE FOR FORMAT
                             <1> ;           DI     = DRIVE #
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS REFLECTS STATUS
                             <1> ;           AH = @DSKETTE_STATUS
                             <1> ;           CY = 1 IF ERROR
                             <1> ;-------------------------------------------------------------------------------
                             <1> FORMAT_SET:
000036D7 E884010000          <1>       CALL  XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
000036DC 6656                <1>       PUSH  SI                  ; SAVE DASD TYPE
000036DE 6689F0              <1>       MOV   AX,SI               ; AH = ? , AL , DASD TYPE
000036E1 30E4                <1>       XOR   AH,AH               ; AH , 0 , AL , DASD TYPE
000036E3 6689C6              <1>       MOV   SI,AX               ; SI = DASD TYPE
000036E6 80A7[CD580100]0F    <1>       AND   byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
000036ED 664E                <1>       DEC   SI                  ; CHECK FOR 320/360K MEDIA & DRIVE
000036EF 7509                <1>       JNZ   short NOT_320       ; BYPASS IF NOT
000036F1 808F[CD580100]90    <1>       OR    byte [DSK_STATE+eDI], MED_DET+RATE_250 ; SET TO 320/360
000036F8 EB48                <1>       JMP   SHORT S0
                             <1>
                             <1> NOT_320:
000036FA E8B6030000          <1>       CALL  MED_CHANGE          ; CHECK FOR TIME_OUT
000036FF 803D[C0580100]80    <1>       CMP   byte [DSKETTE_STATUS], TIME_OUT
00003706 743A                <1>       JZ    short S0            ; IF TIME OUT TELL CALLER
                             <1> S3:
00003708 664E                <1>       DEC   SI                  ; CHECK FOR 320/360K IN 1.2M DRIVE
0000370A 7509                <1>       JNZ   short NOT_320_12    ; BYPASS IF NOT
0000370C 808F[CD580100]70    <1>       OR    byte [DSK_STATE+eDI], MED_DET+DBL_STEP+RATE_300 ; SET STATE
00003713 EB2D                <1>       JMP   SHORT S0
                             <1>
                             <1> NOT_320_12:
00003715 664E                <1>       DEC   SI                  ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
00003717 7509                <1>       JNZ   short NOT_12        ; BYPASS IF NOT
00003719 808F[CD580100]10    <1>       OR    byte [DSK_STATE+eDI], MED_DET+RATE_500 ; SET STATE VARIABLE
```

```
00003720 EB20                 <1>       JMP    SHORT S0            ; RETURN TO CALLER
                              <1>
                              <1> NOT_12:
00003722 664E                 <1>       DEC    SI                  ; CHECK FOR SET DASD TYPE 04
00003724 752B                 <1>       JNZ    short FS_ERR        ; BAD COMMAND EXIT IF NOT VALID TYPE
                              <1>
00003726 F687[CD580100]04     <1>       TEST   byte [DSK_STATE+eDI], DRV_DET ; DRIVE DETERMINED ?
0000372D 740B                 <1>       JZ     short ASSUME        ; IF STILL NOT DETERMINED ASSUME
0000372F B050                 <1>       MOV    AL,MED_DET+RATE_300
00003731 F687[CD580100]02     <1>          TEST    byte [DSK_STATE+eDI], FMT_CAPA ; MULTIPLE FORMAT CAPABILITY ?
00003738 7502                 <1>       JNZ    short OR_IT_IN             ; IF 1.2 M THEN DATA RATE 300
                              <1>
                              <1> ASSUME:
0000373A B090                 <1>       MOV    AL,MED_DET+RATE_250 ; SET UP
                              <1>
                              <1> OR_IT_IN:
0000373C 0887[CD580100]       <1>       OR     [DSK_STATE+eDI], AL ; OR IN THE CORRECT STATE
                              <1> S0:
00003742 E84A010000           <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
00003747 E884060000           <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
0000374C 665B                 <1>       POP    BX                  ; GET SAVED AL TO BL
0000374E 88D8                 <1>       MOV    AL,BL               ; PUT BACK FOR RETURN
00003750 C3                   <1>       RETn
                              <1>
                              <1> FS_ERR:
00003751 C605[C0580100]01     <1>       MOV    byte [DSKETTE_STATUS], BAD_CMD ; UNKNOWN STATE,BAD COMMAND
00003758 EBE8                 <1>       JMP    SHORT S0
                              <1>
                              <1> ;-------------------------------------------------------------------------------
                              <1> ; SET_MEDIA  (AH = 18H)
                              <1> ;     THIS ROUTINE SETS THE TYPE OF MEDIA AND DATA RATE
                              <1> ;     TO BE USED FOR THE FOLLOWING FORMAT OPERATION.
                              <1> ;
                              <1> ; ON ENTRY:
                              <1> ;     [BP]   = SECTOR PER TRACK
                              <1> ;     [BP+1] = TRACK #
                              <1> ;     DI     = DRIVE #
                              <1> ;
                              <1> ; ON EXIT:
                              <1> ;     @DSKETTE_STATUS REFLECTS STATUS
                              <1> ;     IF NO ERROR:
                              <1> ;           AH = 0
                              <1> ;           CY = 0
                              <1> ;           ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
                              <1> ;           DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE
                              <1> ;     IF ERROR:
                              <1> ;           AH = @DSKETTE_STATUS
                              <1> ;           CY = 1
                              <1> ;-------------------------------------------------------------------------------
                              <1> SET_MEDIA:
0000375A E801010000           <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
0000375F F687[CD580100]01     <1>          TEST    byte [DSK_STATE+eDI], TRK_CAPA ; CHECK FOR CHANGE LINE AVAILABLE
00003766 7415                 <1>       JZ     short SM_CMOS       ; JUMP IF 40 TRACK DRIVE
00003768 E848030000           <1>       CALL   MED_CHANGE          ; RESET CHANGE LINE
0000376D 803D[C0580100]80     <1>       CMP    byte [DSKETTE_STATUS], TIME_OUT ; IF TIME OUT TELL CALLER
00003774 746B                 <1>       JE     short SM_RTN
00003776 C605[C0580100]00     <1>       MOV    byte [DSKETTE_STATUS], 0 ; CLEAR STATUS
                              <1> SM_CMOS:
0000377D E819070000           <1>       CALL   CMOS_TYPE           ; RETURN DRIVE TYPE IN (AL)
                              <1>       ;;20/02/2015
                              <1>       ;;JC   short MD_NOT_FND    ; ERROR IN CMOS
                              <1>       ;;OR   AL,AL               ; TEST FOR NO DRIVE
00003782 745D                 <1>       JZ     short SM_RTN        ; RETURN IF SO
00003784 E863000000           <1>       CALL   DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
00003789 7231                 <1>       JC     short MD_NOT_FND    ; TYPE NOT IN TABLE (BAD CMOS)
0000378B 57                   <1>       PUSH   eDI                 ; SAVE REG.
0000378C 31DB                 <1>       XOR    eBX,eBX                    ; BX = INDEX TO DR. TYPE TABLE
0000378E B906000000           <1>       MOV    eCX,DR_CNT          ; CX = LOOP COUNT
                              <1> DR_SEARCH:
00003793 8AA3[705C0000]       <1>       MOV    AH, [DR_TYPE+eBX]   ; GET DRIVE TYPE
00003799 80E47F               <1>       AND    AH,BIT7OFF          ; MASK OUT MSB
0000379C 38E0                 <1>       CMP    AL,AH               ; DRIVE TYPE MATCH ?
0000379E 7516                 <1>       JNE    short NXT_MD        ; NO, CHECK NEXT DRIVE TYPE
                              <1> DR_FND:
000037A0 8BBB[715C0000]       <1>       MOV    eDI, [DR_TYPE+eBX+1]       ; DI = MEDIA/DRIVE PARAM TABLE
                              <1> MD_SEARCH:
000037A6 8A6704               <1>          MOV     AH, [eDI+MD.SEC_TRK]    ; GET SECTOR/TRACK
000037A9 386500               <1>       CMP    [eBP],AH            ; MATCH?
000037AC 7508                 <1>       JNE    short NXT_MD        ; NO, CHECK NEXT MEDIA
000037AE 8A670B               <1>          MOV     AH, [eDI+MD.MAX_TRK]    ; GET MAX. TRACK #
000037B1 386501               <1>       CMP    [eBP+1],AH          ; MATCH?
000037B4 740F                 <1>       JE     short MD_FND        ; YES, GO GET RATE
                              <1> NXT_MD:
                              <1>       ;ADD   BX,3                ; CHECK NEXT DRIVE TYPE
000037B6 83C305               <1>          add ebx, 5 ; 18/02/2015
000037B9 E2D8                 <1>       LOOP    DR_SEARCH
000037BB 5F                   <1>       POP    eDI                 ; RESTORE REG.
                              <1> MD_NOT_FND:
000037BC C605[C0580100]0C     <1>       MOV    byte [DSKETTE_STATUS], MED_NOT_FND ; ERROR, MEDIA TYPE NOT FOUND
000037C3 EB1C                 <1>       JMP    SHORT SM_RTN        ; RETURN
                              <1> MD_FND:
000037C5 8A470C               <1>          MOV     AL, [eDI+MD.RATE]       ; GET RATE
000037C8 3C40                 <1>       CMP    AL,RATE_300         ; DOUBLE STEP REQUIRED FOR RATE 300
000037CA 7502                 <1>       JNE    short MD_SET
000037CC 0C20                 <1>       OR     AL,DBL_STEP
                              <1> MD_SET:
                              <1>       ;MOV   [BP+6],DI           ; SAVE TABLE POINTER IN STACK
000037CE 897D0C               <1>       mov    [ebp+12], edi ; 18/02/2015
000037D1 0C10                 <1>       OR     AL,MED_DET          ; SET MEDIA ESTABLISHED
000037D3 5F                   <1>       POP    eDI
000037D4 80A7[CD580100]0F     <1>       AND    byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
000037DB 0887[CD580100]       <1>       OR     [DSK_STATE+eDI], AL
                              <1>       ;MOV   AX, CS              ; SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
```

```
                             <1>       ;MOV   ES, AX              ; ES IS SEGMENT OF TABLE
                             <1> SM_RTN:
000037E1 E8AB000000          <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
000037E6 E8E5050000          <1>       CALL   SETUP_END           ; VARIOUS CLEANUPS
000037EB C3                  <1>       RETn
                             <1>
                             <1> ;---------------------------------------------------------------
                             <1> ; DR_TYPE_CHECK                                            :
                             <1> ;     CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL)              :
                             <1> ;     IS SUPPORTED IN BIOS DRIVE TYPE TABLE                :
                             <1> ; ON ENTRY:                                         :
                             <1> ;     AL = DRIVE TYPE                                      :
                             <1> ; ON EXIT:                                          :
                             <1> ;     CS = SEGMENT MEDIA/DRIVE PARAMETER TABLE (CODE)             :
                             <1> ;     CY = 0       DRIVE TYPE SUPPORTED                           :
                             <1> ;          BX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE             :
                             <1> ;     CY = 1 DRIVE TYPE NOT SUPPORTED               :
                             <1> ; REGISTERS ALTERED: eBX                                   :
                             <1> ;---------------------------------------------------------------
                             <1> DR_TYPE_CHECK:
000037EC 6650                <1>       PUSH   AX
000037EE 51                  <1>       PUSH   eCX
000037EF 31DB                <1>       XOR    eBX,eBX                   ; BX = INDEX TO DR_TYPE TABLE
000037F1 B906000000          <1>       MOV    eCX,DR_CNT          ; CX = LOOP COUNT
                             <1> TYPE_CHK:
000037F6 8AA3[705C0000]      <1>       MOV    AH,[DR_TYPE+eBX]    ; GET DRIVE TYPE
000037FC 38E0                <1>       CMP    AL,AH               ; DRIVE TYPE MATCH?
000037FE 740D                <1>       JE     short DR_TYPE_VALID ; YES, RETURN WITH CARRY RESET
                             <1>       ;ADD   BX,3                ; CHECK NEXT DRIVE TYPE
00003800 83C305              <1>         add ebx, 5 ; 16/02/2015 (32 bit address modification)
00003803 E2F1                <1>       LOOP    TYPE_CHK
                             <1>       ;
00003805 BB[CF5C0000]        <1>       mov    ebx, MD_TBL6        ; 1.44MB fd parameter table
                             <1>                                  ; Default for GET_PARM (11/12/2014)
                             <1>       ;
0000380A F9                  <1>       STC                        ; DRIVE TYPE NOT FOUND IN TABLE
0000380B EB06                <1>       JMP    SHORT TYPE_RTN
                             <1> DR_TYPE_VALID:
0000380D 8B9B[715C0000]      <1>       MOV    eBX,[DR_TYPE+eBX+1]       ; BX = MEDIA TABLE
                             <1> TYPE_RTN:
00003813 59                  <1>       POP    eCX
00003814 6658                <1>       POP    AX
00003816 C3                  <1>       RETn
                             <1>
                             <1> ;---------------------------------------------------------------
                             <1> ; SEND_SPEC                                         :
                             <1> ;     SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM      :
                             <1> ;     THE DRIVE PARAMETER TABLE POINTED BY @DISK_POINTER   :
                             <1> ; ON ENTRY: @DISK_POINTER = DRIVE PARAMETER TABLE          :
                             <1> ; ON EXIT:  NONE                                    :
                             <1> ; REGISTERS ALTERED: CX, DX                                :
                             <1> ;---------------------------------------------------------------
                             <1> SEND_SPEC:
00003817 50                  <1>       PUSH   eAX                 ; SAVE AX
00003818 B8[3E380000]        <1>       MOV    eAX, SPECBAC        ; LOAD ERROR ADDRESS
0000381D 50                  <1>       PUSH   eAX                 ; PUSH NEC_OUT ERROR RETURN
0000381E B403                <1>       MOV    AH,03H              ; SPECIFY COMMAND
00003820 E885070000          <1>       CALL   NEC_OUTPUT          ; OUTPUT THE COMMAND
00003825 28D2                <1>       SUB    DL,DL               ; FIRST SPECIFY BYTE
00003827 E878060000          <1>       CALL   GET_PARM            ; GET PARAMETER TO AH
0000382C E879070000          <1>       CALL   NEC_OUTPUT          ; OUTPUT THE COMMAND
00003831 B201                <1>       MOV    DL,1                ; SECOND SPECIFY BYTE
00003833 E86C060000          <1>       CALL   GET_PARM            ; GET PARAMETER TO AH
00003838 E86D070000          <1>       CALL   NEC_OUTPUT          ; OUTPUT THE COMMAND
0000383D 58                  <1>       POP    eAX                 ; POP ERROR RETURN
                             <1> SPECBAC:
0000383E 58                  <1>       POP    eAX                 ; RESTORE ORIGINAL AX VALUE
0000383F C3                  <1>       RETn
                             <1>
                             <1> ;---------------------------------------------------------------
                             <1> ; SEND_SPEC_MD                                             :
                             <1> ;     SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM      :
                             <1> ;     THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX)   :
                             <1> ; ON ENTRY: CS:BX = MEDIA/DRIVE PARAMETER TABLE            :
                             <1> ; ON EXIT:  NONE                                    :
                             <1> ; REGISTERS ALTERED: AX                                    :
                             <1> ;---------------------------------------------------------------
                             <1> SEND_SPEC_MD:
00003840 50                  <1>       PUSH   eAX                 ; SAVE RATE DATA
00003841 B8[5E380000]        <1>       MOV    eAX, SPEC_ESBAC            ; LOAD ERROR ADDRESS
00003846 50                  <1>       PUSH   eAX                 ; PUSH NEC_OUT ERROR RETURN
00003847 B403                <1>       MOV    AH,03H              ; SPECIFY COMMAND
00003849 E85C070000          <1>       CALL   NEC_OUTPUT          ; OUTPUT THE COMMAND
0000384E 8A23                <1>         MOV     AH, [eBX+MD.SPEC1]       ; GET 1ST SPECIFY BYTE
00003850 E855070000          <1>       CALL   NEC_OUTPUT          ; OUTPUT THE COMMAND
00003855 8A6301              <1>         MOV     AH, [eBX+MD.SPEC2]       ; GET SECOND SPECIFY BYTE
00003858 E84D070000          <1>       CALL   NEC_OUTPUT          ; OUTPUT THE COMMAND
0000385D 58                  <1>       POP    eAX                 ; POP ERROR RETURN
                             <1> SPEC_ESBAC:
0000385E 58                  <1>       POP    eAX                 ; RESTORE ORIGINAL AX VALUE
0000385F C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; XLAT_NEW
                             <1> ;     TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE
                             <1> ;     MODE TO NEW ARCHITECTURE.
                             <1> ;
                             <1> ; ON ENTRY: DI = DRIVE #
                             <1> ;-------------------------------------------------------------------------------
                             <1> XLAT_NEW:
00003860 83FF01              <1>       CMP    eDI,1                     ; VALID DRIVE
00003863 7725                <1>       JA     short XN_OUT              ; IF INVALID BACK
00003865 80BF[CD580100]00    <1>       CMP    byte [DSK_STATE+eDI], 0          ; NO DRIVE ?
```

```
0000386C 741D                <1>       JZ     short DO_DET              ; IF NO DRIVE ATTEMPT DETERMINE
0000386E 6689F9              <1>       MOV    CX,DI                     ; CX = DRIVE NUMBER
00003871 C0E102              <1>       SHL    CL,2                      ; CL = SHIFT COUNT, A=0, B=4
00003874 A0[CC580100]        <1>       MOV    AL, [HF_CNTRL]                   ; DRIVE INFORMATION
00003879 D2C8                <1>       ROR    AL,CL                     ; TO LOW NIBBLE
0000387B 2407                <1>       AND    AL,DRV_DET+FMT_CAPA+TRK_CAPA     ; KEEP DRIVE BITS
0000387D 80A7[CD580100]F8    <1>         AND     byte [DSK_STATE+eDI], ~(DRV_DET+FMT_CAPA+TRK_CAPA)
00003884 0887[CD580100]      <1>       OR     [DSK_STATE+eDI], AL       ; UPDATE DRIVE STATE
                             <1> XN_OUT:
0000388A C3                  <1>       RETn
                             <1> DO_DET:
0000388B E8BF080000          <1>       CALL   DRIVE_DET                 ; TRY TO DETERMINE
00003890 C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; XLAT_OLD
                             <1> ;     TRANSLATES DISKETTE STATE LOCATIONS FROM NEW
                             <1> ;     ARCHITECTURE TO COMPATIBLE MODE.
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE
                             <1> ;-------------------------------------------------------------------------------
                             <1> XLAT_OLD:
00003891 83FF01              <1>       CMP    eDI,1              ; VALID DRIVE ?
                             <1>         ;JA     short XO_OUT            ; IF INVALID BACK
00003894 0F8786000000        <1>         ja      XO_OUT
0000389A 80BF[CD580100]00    <1>         CMP  byte [DSK_STATE+eDI],0    ; NO DRIVE ?
000038A1 747D                <1>       JZ     short XO_OUT       ; IF NO DRIVE TRANSLATE DONE
                             <1>
                             <1> ;-----      TEST FOR SAVED DRIVE INFORMATION ALREADY SET
                             <1>
000038A3 6689F9              <1>       MOV    CX,DI              ; CX = DRIVE NUMBER
000038A6 C0E102              <1>       SHL    CL,2               ; CL = SHIFT COUNT, A=0, B=4
000038A9 B402                <1>       MOV    AH,FMT_CAPA        ; LOAD MULTIPLE DATA RATE BIT MASK
000038AB D2CC                <1>       ROR    AH,CL              ; ROTATE BY MASK
000038AD 8425[CC580100]      <1>       TEST   [HF_CNTRL], AH            ; MULTIPLE-DATA RATE DETERMINED ?
000038B3 751C                <1>       JNZ    short SAVE_SET            ; IF SO, NO NEED TO RE-SAVE
                             <1>
                             <1> ;-----      ERASE DRIVE BITS IN @HF_CNTRL FOR THIS DRIVE
                             <1>
000038B5 B407                <1>       MOV    AH,DRV_DET+FMT_CAPA+TRK_CAPA ; MASK TO KEEP
000038B7 D2CC                <1>       ROR    AH,CL              ; FIX MASK TO KEEP
000038B9 F6D4                <1>       NOT    AH                 ; TRANSLATE MASK
000038BB 2025[CC580100]      <1>       AND    [HF_CNTRL], AH            ; KEEP BITS FROM OTHER DRIVE INTACT
                             <1>
                             <1> ;-----      ACCESS CURRENT DRIVE BITS AND STORE IN @HF_CNTRL
                             <1>
000038C1 8A87[CD580100]      <1>       MOV    AL, [DSK_STATE+eDI] ; ACCESS STATE
000038C7 2407                <1>       AND    AL,DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
000038C9 D2C8                <1>       ROR    AL,CL              ; FIX FOR THIS DRIVE
000038CB 0805[CC580100]      <1>       OR     [HF_CNTRL], AL            ; UPDATE SAVED DRIVE STATE
                             <1>
                             <1> ;-----      TRANSLATE TO COMPATIBILITY MODE
                             <1>
                             <1> SAVE_SET:
000038D1 8AA7[CD580100]      <1>       MOV    AH, [DSK_STATE+eDI] ; ACCESS STATE
000038D7 88E7                <1>       MOV    BH,AH              ; TO BH FOR LATER
000038D9 80E4C0              <1>       AND    AH,RATE_MSK        ; KEEP ONLY RATE
000038DC 80FC00              <1>       CMP    AH,RATE_500        ; RATE 500 ?
000038DF 7410                <1>       JZ     short CHK_144      ; YES 1.2/1.2 OR 1.44/1.44
000038E1 B001                <1>       MOV    AL,M3D1U           ; AL = 360 IN 1.2 UNESTABLISHED
000038E3 80FC40              <1>       CMP    AH,RATE_300        ; RATE 300 ?
000038E6 7518                <1>       JNZ    short CHK_250      ; NO, 360/360, 720/720 OR 720/1.44
000038E8 F6C720              <1>       TEST   BH,DBL_STEP        ; CHECK FOR DOUBLE STEP
000038EB 751F                <1>       JNZ    short TST_DET      ; MUST BE 360 IN 1.2
                             <1> UNKNO:
000038ED B007                <1>       MOV    AL,MED_UNK         ; NONE OF THE ABOVE
000038EF EB22                <1>       JMP    SHORT AL_SET       ; PROCESS COMPLETE
                             <1> CHK_144:
000038F1 E8A5050000          <1>       CALL   CMOS_TYPE          ; RETURN DRIVE TYPE IN (AL)
                             <1>       ;;20/02/2015
                             <1>       ;;JC   short UNKNO        ; ERROR, SET 'NONE OF ABOVE'
000038F6 74F5                <1>       jz     short UNKNO ;; 20/02/2015
000038F8 3C02                <1>       CMP    AL,2               ; 1.2MB DRIVE ?
000038FA 75F1                <1>       JNE    short UNKNO        ; NO, GO SET 'NONE OF ABOVE'
000038FC B002                <1>       MOV    AL,M1D1U           ; AL = 1.2 IN 1.2 UNESTABLISHED
000038FE EB0C                <1>       JMP    SHORT TST_DET
                             <1> CHK_250:
00003900 B000                <1>       MOV    AL,M3D3U           ; AL = 360 IN 360 UNESTABLISHED
00003902 80FC80              <1>       CMP    AH,RATE_250        ; RATE 250 ?
00003905 75E6                <1>       JNZ    short UNKNO        ; IF SO FALL IHRU
00003907 F6C701              <1>       TEST   BH,TRK_CAPA        ; 80 TRACK CAPABILITY ?
0000390A 75E1                <1>       JNZ    short UNKNO        ; IF SO JUMP, FALL THRU TEST DET
                             <1> TST_DET:
0000390C F6C710              <1>       TEST   BH,MED_DET         ; DETERMINED ?
0000390F 7402                <1>       JZ     short AL_SET       ; IF NOT THEN SET
00003911 0403                <1>       ADD    AL,3               ; MAKE DETERMINED/ESTABLISHED
                             <1> AL_SET:
00003913 80A7[CD580100]F8    <1>       AND    byte [DSK_STATE+eDI], ~(DRV_DET+FMT_CAPA+TRK_CAPA) ; CLEAR DRIVE
0000391A 0887[CD580100]      <1>       OR     [DSK_STATE+eDI], AL ; REPLACE WITH COMPATIBLE MODE
                             <1> XO_OUT:
00003920 C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; RD_WR_VF
                             <1> ;     COMMON READ, WRITE AND VERIFY:
                             <1> ;     MAIN LOOP FOR STATE RETRIES.
                             <1> ;
                             <1> ; ON ENTRY:  AH = READ/WRITE/VERIFY NEC PARAMETER
                             <1> ;            AL = READ/WRITE/VERIFY DMA PARAMETER
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;-------------------------------------------------------------------------------
                             <1> RD_WR_VF:
```

```
00003921 6650                <1>       PUSH  AX                  ; SAVE DMA, NEC PARAMETERS
00003923 E838FFFFFF          <1>       CALL  XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH.
00003928 E8F3000000          <1>       CALL  SETUP_STATE         ; INITIALIZE START AND END RATE
0000392D 6658                <1>       POP   AX                  ; RESTORE READ/WRITE/VERIFY
                             <1> DO_AGAIN:
0000392F 6650                <1>       PUSH  AX                  ; SAVE READ/WRITE/VERIFY PARAMETER
00003931 E87F010000          <1>       CALL  MED_CHANGE          ; MEDIA CHANGE AND RESET IF CHANGED
00003936 6658                <1>       POP   AX                  ; RESTORE READ/WRITE/VERIFY
00003938 0F82C9000000        <1>         JC      RWV_END                 ; MEDIA CHANGE ERROR OR TIME-OUT
                             <1> RWV:
0000393E 6650                <1>       PUSH  AX                  ; SAVE READ/WRITE/VERIFY PARAMETER
00003940 8AB7[CD580100]      <1>       MOV   DH, [DSK_STATE+eDI] ; GET RATE STATE OF THIS DRIVE
00003946 80E6C0              <1>       AND   DH,RATE_MSK         ; KEEP ONLY RATE
00003949 E84D050000          <1>       CALL  CMOS_TYPE           ; RETURN DRIVE TYPE IN AL (AL)
                             <1>       ;;20/02/2015
                             <1>       ;;JC  short RWV_ASSUME    ; ERROR IN CMOS
0000394E 7451                <1>       jz    short RWV_ASSUME ; 20/02/2015
00003950 3C01                <1>       CMP   AL,1                ; 40 TRACK DRIVE?
00003952 750D                <1>       JNE   short RWV_1         ; NO, BYPASS CMOS VALIDITY CHECK
00003954 F687[CD580100]01    <1>       TEST  byte [DSK_STATE+eDI], TRK_CAPA ; CHECK FOR 40 TRACK DRIVE
0000395B 7413                <1>       JZ    short RWV_2         ; YES, CMOS IS CORRECT
0000395D B002                <1>       MOV   AL,2                ; CHANGE TO 1.2M
0000395F EB0F                <1>       JMP   SHORT RWV_2
                             <1> RWV_1:
00003961 720D                <1>       JB    short RWV_2         ; NO DRIVE SPECIFIED, CONTINUE
00003963 F687[CD580100]01    <1>       TEST   byte [DSK_STATE+eDI], TRK_CAPA ; IS IT REALLY 40 TRACK?
0000396A 7504                <1>       JNZ   short RWV_2         ; NO, 80 TRACK
0000396C B001                <1>       MOV   AL,1                ; IT IS 40 TRACK, FIX CMOS VALUE
0000396E EB04                <1>       jmp   short rwv_3
                             <1> RWV_2:
00003970 08C0                <1>       OR    AL,AL               ; TEST FOR NO DRIVE
00003972 742D                <1>       JZ    short RWV_ASSUME    ; ASSUME TYPE, USE MAX TRACK
                             <1> rwv_3:
00003974 E873FEFFFF          <1>       CALL  DR_TYPE_CHECK       ; RTN CS:BX = MEDIA/DRIVE PARAM TBL.
00003979 7226                <1>       JC    short RWV_ASSUME    ; TYPE NOT IN TABLE (BAD CMOS)
                             <1>
                             <1> ;-----      SEARCH FOR MEDIA/DRIVE PARAMETER TABLE
                             <1>
0000397B 57                  <1>       PUSH  eDI                 ; SAVE DRIVE #
0000397C 31DB                <1>       XOR   eBX,eBX                   ; BX = INDEX TO DR_TYPE TABLE
0000397E B906000000          <1>       MOV   eCX,DR_CNT          ; CX = LOOP COUNT
                             <1> RWV_DR_SEARCH:
00003983 8AA3[705C0000]      <1>       MOV   AH, [DR_TYPE+eBX]   ; GET DRIVE TYPE
00003989 80E47F              <1>       AND   AH,BIT7OFF          ; MASK OUT MSB
0000398C 38E0                <1>       CMP   AL,AH               ; DRIVE TYPE MATCH?
0000398E 750B                <1>       JNE   short RWV_NXT_MD    ; NO, CHECK NEXT DRIVE TYPE
                             <1> RWV_DR_FND:
00003990 8BBB[715C0000]      <1>       MOV   eDI, [DR_TYPE+eBX+1]      ; DI = MEDIA/DRIVE PARAMETER TABLE
                             <1> RWV_MD_SEARH:
00003996 3A770C              <1>         CMP     DH, [eDI+MD.RATE]       ; MATCH?
00003999 741B                <1>       JE    short RWV_MD_FND    ; YES, GO GET 1ST SPECIFY BYTE
                             <1> RWV_NXT_MD:
                             <1>       ;ADD  BX,3                ; CHECK NEXT DRIVE TYPE
0000399B 83C305              <1>       add   eBX, 5
0000399E E2E3                <1>       LOOP  RWV_DR_SEARCH
000039A0 5F                  <1>       POP   eDI                 ; RESTORE DRIVE #
                             <1>
                             <1> ;-----      ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED
                             <1>
                             <1> RWV_ASSUME:
000039A1 BB[8E5C0000]        <1>       MOV   eBX, MD_TBL1        ; POINT TO 40 TRACK 250 KBS
000039A6 F687[CD580100]01    <1>       TEST  byte [DSK_STATE+eDI], TRK_CAPA ; TEST FOR 80 TRACK
000039AD 740A                <1>       JZ    short RWV_MD_FND1   ; MUST BE 40 TRACK
000039AF BB[A85C0000]        <1>       MOV   eBX, MD_TBL3        ; POINT TO 80 TRACK 500 KBS
000039B4 EB03                <1>       JMP   short RWV_MD_FND1   ; GO SPECIFY PARAMTERS
                             <1>
                             <1> ;-----      CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE
                             <1>
                             <1> RWV_MD_FND:
000039B6 89FB                <1>       MOV   eBX,eDI                   ; BX = MEDIA/DRIVE PARAMETER TABLE
000039B8 5F                  <1>       POP   eDI                 ; RESTORE DRIVE #
                             <1>
                             <1> ;-----      SEND THE SPECIFY COMMAND TO THE CONTROLLER
                             <1>
                             <1> RWV_MD_FND1:
000039B9 E882FEFFFF          <1>       CALL  SEND_SPEC_MD
000039BE E864010000          <1>       CALL  CHK_LASTRATE        ; ZF=1 ATTEMP RATE IS SAME AS LAST RATE
000039C3 7405                <1>       JZ    short RWV_DBL       ; YES,SKIP SEND RATE COMMAND
000039C5 E83B010000          <1>       CALL  SEND_RATE           ; SEND DATA RATE TO NEC
                             <1> RWV_DBL:
000039CA 53                  <1>       PUSH  eBX                 ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
000039CB E822040000          <1>       CALL  SETUP_DBL           ; CHECK FOR DOUBLE STEP
000039D0 5B                  <1>       POP   eBX                 ; RESTORE ADDRESS
000039D1 7226                <1>       JC    short CHK_RET       ; ERROR FROM READ ID, POSSIBLE RETRY
000039D3 6658                <1>       POP   AX                  ; RESTORE NEC, DMA COMMAND
000039D5 6650                <1>       PUSH  AX                  ; SAVE NEC COMMAND
000039D7 53                  <1>       PUSH  eBX                 ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
000039D8 E861010000          <1>       CALL  DMA_SETUP           ; SET UP THE DMA
000039DD 5B                  <1>       POP   eBX
000039DE 6658                <1>       POP   AX                  ; RESTORE NEC COMMAND
000039E0 722F                <1>       JC    short RWV_BAC       ; CHECK FOR DMA BOUNDARY ERROR
000039E2 6650                <1>       PUSH  AX                  ; SAVE NEC COMMAND
000039E4 53                  <1>       PUSH  eBX                 ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
000039E5 E83C020000          <1>       CALL  NEC_INIT            ; INITIALIZE NEC
000039EA 5B                  <1>       POP   eBX                 ; RESTORE ADDRESS
000039EB 720C                <1>       JC    short CHK_RET       ; ERROR - EXIT
000039ED E866020000          <1>       CALL  RWV_COM                   ; OP CODE COMMON TO READ/WRITE/VERIFY
000039F2 7205                <1>       JC    short CHK_RET       ; ERROR - EXIT
000039F4 E8AB020000          <1>       CALL  NEC_TERM            ; TERMINATE, GET STATUS, ETC.
                             <1> CHK_RET:
000039F9 E84A030000          <1>       CALL  RETRY               ; CHECK FOR, SETUP RETRY
000039FE 6658                <1>       POP   AX                  ; RESTORE READ/WRITE/VERIFY PARAMETER
00003A00 7305                <1>       JNC   short RWV_END       ; CY = 0 NO RETRY
```

```
00003A02 E928FFFFFF          <1>         JMP     DO_AGAIN                ; CY = 1 MEANS RETRY
                             <1> RWV_END:
00003A07 E8F4020000          <1>       CALL   DSTATE             ; ESTABLISH STATE IF SUCCESSFUL
00003A0C E887030000          <1>       CALL   NUM_TRANS          ; AL = NUMBER TRANSFERRED
                             <1> RWV_BAC:                        ; BAD DMA ERROR ENTRY
00003A11 6650                <1>       PUSH   AX                 ; SAVE NUMBER TRANSFERRED
00003A13 E879FEFFFF          <1>       CALL   XLAT_OLD           ; TRANSLATE STATE TO COMPATIBLE MODE
00003A18 6658                <1>       POP    AX                 ; RESTORE NUMBER TRANSFERRED
00003A1A E8B1030000          <1>       CALL   SETUP_END          ; VARIOUS CLEANUPS
00003A1F C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; SETUP_STATE:     INITIALIZES START AND END RATES.
                             <1> ;-------------------------------------------------------------------------------
                             <1> SETUP_STATE:
00003A20 F687[CD580100]10    <1>       TEST   byte [DSK_STATE+eDI], MED_DET ; MEDIA DETERMINED ?
00003A27 7537                <1>       JNZ    short J1C          ; NO STATES IF DETERMINED
00003A29 66B84000            <1>         MOV     AX,(RATE_500*256)+RATE_300  ; AH = START RATE, AL = END RATE
00003A2D F687[CD580100]04    <1>       TEST   byte [DSK_STATE+eDI],DRV_DET ; DRIVE ?
00003A34 740D                <1>       JZ     short AX_SET       ; DO NOT KNOW DRIVE
00003A36 F687[CD580100]02    <1>       TEST   byte [DSK_STATE+eDI], FMT_CAPA ; MULTI-RATE?
00003A3D 7504                <1>       JNZ    short AX_SET       ; JUMP IF YES
00003A3F 66B88080            <1>         MOV     AX,RATE_250*257         ; START A END RATE 250 FOR 360 DRIVE
                             <1> AX_SET:
00003A43 80A7[CD580100]1F    <1>       AND    byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP) ; TURN OFF THE RATE
00003A4A 08A7[CD580100]      <1>       OR     [DSK_STATE+eDI], AH ; RATE FIRST TO TRY
00003A50 8025[C8580100]F3    <1>       AND    byte [LASTRATE], ~STRT_MSK ; ERASE LAST TO TRY RATE BITS
00003A57 C0C804              <1>       ROR    AL,4               ; TO OPERATION LAST RATE LOCATION
00003A5A 0805[C8580100]      <1>       OR     [LASTRATE], AL            ; LAST RATE
                             <1> J1C:
00003A60 C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ;  FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
                             <1> ;-------------------------------------------------------------------------------
                             <1> FMT_INIT:
00003A61 F687[CD580100]10    <1>       TEST   byte [DSK_STATE+eDI], MED_DET ; IS MEDIA ESTABLISHED
00003A68 7546                <1>       JNZ    short F1_OUT       ; IF SO RETURN
00003A6A E82C040000          <1>       CALL   CMOS_TYPE          ; RETURN DRIVE TYPE IN AL
                             <1>       ;; 20/02/2015
                             <1>       ;;JC   short CL_DRV       ; ERROR IN CMOS ASSUME NO DRIVE
00003A6F 7440                <1>       jz     short CL_DRV ;; 20/02/2015
00003A71 FEC8                <1>       DEC    AL                 ; MAKE ZERO ORIGIN
                             <1>       ;;JS   short CL_DRV       ; NO DRIVE IF AL 0
00003A73 8AA7[CD580100]      <1>       MOV    AH, [DSK_STATE+eDI] ; AH = CURRENT STATE
00003A79 80E40F              <1>       AND    AH, ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR
00003A7C 08C0                <1>       OR     AL,AL              ; CHECK FOR 360
00003A7E 7505                <1>       JNZ    short N_360        ; IF 360 WILL BE 0
00003A80 80CC90              <1>       OR     AH,MED_DET+RATE_250 ; ESTABLISH MEDIA
00003A83 EB25                <1>       JMP    SHORT SKP_STATE           ; SKIP OTHER STATE PROCESSING
                             <1> N_360:
00003A85 FEC8                <1>       DEC    AL                 ; 1.2 M DRIVE
00003A87 7505                <1>       JNZ    short N_12         ; JUMP IF NOT
                             <1> F1_RATE:
00003A89 80CC10              <1>       OR     AH,MED_DET+RATE_500 ; SET FORMAT RATE
00003A8C EB1C                <1>       JMP    SHORT SKP_STATE           ; SKIP OTHER STATE PROCESSING
                             <1> N_12:
00003A8E FEC8                <1>       DEC    AL                 ; CHECK FOR TYPE 3
00003A90 750F                <1>       JNZ    short N_720        ; JUMP IF NOT
00003A92 F6C404              <1>       TEST   AH,DRV_DET         ; IS DRIVE DETERMINED
00003A95 7410                <1>       JZ     short ISNT_12      ; TREAT AS NON 1.2 DRIVE
00003A97 F6C402              <1>       TEST   AH,FMT_CAPA        ; IS 1.2M
00003A9A 740B                <1>       JZ     short ISNT_12      ; JUMP IF NOT
00003A9C 80CC50              <1>       OR     AH,MED_DET+RATE_300 ; RATE 300
00003A9F EB09                <1>       JMP    SHORT SKP_STATE           ; CONTINUE
                             <1> N_720:
00003AA1 FEC8                <1>       DEC    AL                 ; CHECK FOR TYPE 4
00003AA3 750C                <1>       JNZ    short CL_DRV       ; NO DRIVE, CMOS BAD
00003AA5 EBE2                <1>       JMP    SHORT F1_RATE
                             <1> ISNT_12:
00003AA7 80CC90              <1>       OR     AH,MED_DET+RATE_250 ; MUST BE RATE 250
                             <1>
                             <1> SKP_STATE:
00003AAA 88A7[CD580100]      <1>       MOV    [DSK_STATE+eDI], AH ; STORE AWAY
                             <1> F1_OUT:
00003AB0 C3                  <1>       RETn
                             <1> CL_DRV:
00003AB1 30E4                <1>       XOR    AH,AH              ; CLEAR STATE
00003AB3 EBF5                <1>       JMP    SHORT SKP_STATE           ; SAVE IT
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; MED_CHANGE
                             <1> ;     CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,
                             <1> ;     CHECKS MEDIA CHANGE AGAIN.
                             <1> ;
                             <1> ; ON EXIT:   CY = 1 MEANS MEDIA CHANGE OR TIMEOUT
                             <1> ;            @DSKETTE_STATUS = ERROR CODE
                             <1> ;-------------------------------------------------------------------------------
                             <1> MED_CHANGE:
00003AB5 E888060000          <1>       CALL   READ_DSKCHNG       ; READ DISK CHANCE LINE STATE
00003ABA 7447                <1>       JZ     short MC_OUT       ; BYPASS HANDLING DISK CHANGE LINE
00003ABC 80A7[CD580100]EF    <1>       AND    byte [DSK_STATE+eDI], ~MED_DET ; CLEAR STATE FOR THIS DRIVE
                             <1>
                             <1> ;     THIS SEQUENCE ENSURES WHENEVER A DISKETTE IS CHANGED THAT
                             <1> ;     ON THE NEXT OPERATION THE REQUIRED MOTOR START UP TIME WILL
                             <1> ;     BE WAITED. (DRIVE MOTOR MAY GO OFF UPON DOOR OPENING).
                             <1>
00003AC3 6689F9              <1>       MOV    CX,DI              ; CL = DRIVE 0
00003AC6 B001                <1>       MOV    AL,1               ; MOTOR ON BIT MASK
00003AC8 D2E0                <1>       SHL    AL,CL              ; TO APPROPRIATE POSITION
00003ACA F6D0                <1>       NOT    AL                 ; KEEP ALL BUT MOTOR ON
00003ACC FA                  <1>       CLI                       ; NO INTERRUPTS
00003ACD 2005[BE580100]      <1>       AND    [MOTOR_STATUS], AL  ; TURN MOTOR OFF INDICATOR
```

```
00003AD3 FB                  <1>       STI                       ; INTERRUPTS ENABLED
00003AD4 E810040000          <1>       CALL   MOTOR_ON           ; TURN MOTOR ON
                             <1>
                             <1> ;-----       THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL
                             <1>
00003AD9 E86DF9FFFF          <1>       CALL   DSK_RESET          ; RESET NEC
00003ADE B501                <1>       MOV    CH,01H             ; MOVE TO CYLINDER 1
00003AE0 E8FF040000          <1>       CALL   SEEK               ; ISSUE SEEK
00003AE5 30ED                <1>       XOR    CH,CH              ; MOVE TO CYLINDER 0
00003AE7 E8F8040000          <1>       CALL   SEEK               ; ISSUE SEEK
00003AEC C605[C0580100]06    <1>       MOV    byte [DSKETTE_STATUS], MEDIA_CHANGE ; STORE IN STATUS
                             <1> OK1:
00003AF3 E84A060000          <1>       CALL   READ_DSKCHNG       ; CHECK MEDIA CHANGED AGAIN
00003AF8 7407                <1>       JZ     short OK2          ; IF ACTIVE, NO DISKETTE, TIMEOUT
                             <1> OK4:
00003AFA C605[C0580100]80    <1>       MOV    byte [DSKETTE_STATUS], TIME_OUT ; TIMEOUT IF DRIVE EMPTY
                             <1> OK2:
00003B01 F9                  <1>       STC                       ; MEDIA CHANGED, SET CY
00003B02 C3                  <1>       RETn
                             <1> MC_OUT:
00003B03 F8                  <1>       CLC                       ; NO MEDIA CHANGED, CLEAR CY
00003B04 C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; SEND_RATE
                             <1> ;     SENDS DATA RATE COMMAND TO NEC
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ; ON EXIT:   NONE
                             <1> ; REGISTERS ALTERED: DX
                             <1> ;-------------------------------------------------------------------------------
                             <1> SEND_RATE:
00003B05 6650                <1>       PUSH   AX                 ; SAVE REG.
00003B07 8025[C8580100]3F    <1>       AND    byte [LASTRATE], ~SEND_MSK ; ELSE CLEAR LAST RATE ATTEMPTED
00003B0E 8A87[CD580100]      <1>       MOV    AL, [DSK_STATE+eDI] ; GET RATE STATE OF THIS DRIVE
00003B14 24C0                <1>       AND    AL,SEND_MSK        ; KEEP ONLY RATE BITS
00003B16 0805[C8580100]      <1>       OR     [LASTRATE], AL            ; SAVE NEW RATE FOR NEXT CHECK
00003B1C C0C002              <1>       ROL    AL,2               ; MOVE TO BIT OUTPUT POSITIONS
00003B1F 66BAF703            <1>       MOV    DX,03F7H           ; OUTPUT NEW DATA RATE
00003B23 EE                  <1>       OUT    DX,AL
00003B24 6658                <1>       POP    AX                 ; RESTORE REG.
00003B26 C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; CHK_LASTRATE
                             <1> ;     CHECK PREVIOUS DATE RATE SNT TO THE CONTROLLER.
                             <1> ; ON ENTRY:
                             <1> ;     DI = DRIVE #
                             <1> ; ON EXIT:
                             <1> ;     ZF =  1 DATA RATE IS THE SAME AS THE LAST RATE SENT TO NEC
                             <1> ;     ZF =  0 DATA RATE IS DIFFERENT FROM LAST RATE
                             <1> ; REGISTERS ALTERED: DX
                             <1> ;-------------------------------------------------------------------------------
                             <1> CHK_LASTRATE:
00003B27 6650                <1>       PUSH   AX                 ; SAVE REG
00003B29 2225[C8580100]      <1>       AND    AH, [LASTRATE]            ; GET LAST DATA RATE SELECTED
00003B2F 8A87[CD580100]      <1>       MOV    AL, [DSK_STATE+eDI] ; GET RATE STATE OF THIS DRIVE
00003B35 6625C0C0            <1>         AND     AX, SEND_MSK*257        ; KEEP ONLY RATE BITS OF BOTH
00003B39 38E0                <1>       CMP    AL, AH             ; COMPARE TO PREVIOUSLY TRIED
                             <1>                                 ; ZF = 1 RATE IS THE SAME
00003B3B 6658                <1>       POP    AX                 ; RESTORE REG.
00003B3D C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; DMA_SETUP
                             <1> ;     THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY OPERATIONS.
                             <1> ;
                             <1> ; ON ENTRY: AL = DMA COMMAND
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;-------------------------------------------------------------------------------
                             <1>
                             <1> ; SI = Head #, # of Sectors or DASD Type
                             <1>
                             <1> ; 22/08/2015
                             <1> ; 08/02/2015 - Protected Mode Modification
                             <1> ; 06/02/2015 - 07/02/2015
                             <1> ; NOTE: Buffer address must be in 1st 16MB of Physical Memory (24 bit limit).
                             <1> ; (DMA Addres = Physical Address)
                             <1> ; (Retro UNIX 386 v1 Kernel/System Mode Virtual Address = Physical Address)
                             <1> ;
                             <1>
                             <1>
                             <1> ; 04/02/2016 (clc)
                             <1> ; 20/02/2015 modification (source: AWARD BIOS 1999, DMA_SETUP)
                             <1> ; 16/12/2014 (IODELAY)
                             <1>
                             <1> DMA_SETUP:
                             <1>
                             <1> ;; 20/02/2015
00003B3E 8B5504              <1>       mov    edx, [ebp+4]       ; Buffer address
00003B41 F7C2000000FF        <1>       test   edx, 0FF000000h           ; 16 MB limit (22/08/2015, bugfix)
00003B47 756E                <1>       jnz    short dma_bnd_err_stc
                             <1>       ;
00003B49 6650                <1>       push   ax                 ; DMA command
00003B4B 52                  <1>       push   edx                ; *
00003B4C B203                <1>       mov    dl, 3              ; GET BYTES/SECTOR PARAMETER
00003B4E E851030000          <1>       call   GET_PARM           ;
00003B53 88E1                <1>       mov    cl, ah                    ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
00003B55 6689F0              <1>       mov    ax, si             ; Sector count
00003B58 88C4                <1>       mov    ah, al             ; AH =  # OF SECTORS
00003B5A 28C0                <1>       sub    al, al             ; AL = 0, AX = # SECTORS * 256
00003B5C 66D1E8              <1>       shr    ax, 1              ; AX = # SECTORS * 128
00003B5F 66D3E0              <1>       shl    ax, cl             ; SHIFT BY PARAMETER VALUE
```

```
00003B62 6648                <1>       dec    ax                  ; -1 FOR DMA VALUE
00003B64 6689C1              <1>       mov    cx, ax
00003B67 5A                  <1>       pop    edx                 ; *
00003B68 6658                <1>       pop    ax
00003B6A 3C42                <1>       cmp    al, 42h
00003B6C 7507                <1>         jne     short NOT_VERF
00003B6E BA0000FF00          <1>       mov    edx, 0FF0000h
00003B73 EB08                <1>       jmp    short J33
                             <1> NOT_VERF:
00003B75 6601CA              <1>       add    dx, cx              ; check for overflow
00003B78 723E                <1>       jc     short dma_bnd_err
                             <1>       ;
00003B7A 6629CA              <1>       sub    dx, cx              ; Restore start address
                             <1> J33:
00003B7D FA                  <1>       CLI                        ; DISABLE INTERRUPTS DURING DMA SET-UP
00003B7E E60C                <1>       OUT    DMA+12,AL           ; SET THE FIRST/LA5T F/F
                             <1>       IODELAY                           ; WAIT FOR I/O
00003B80 EB00                <2>  jmp short $+2
00003B82 EB00                <2>  jmp short $+2
00003B84 E60B                <1>       OUT    DMA+11,AL           ; OUTPUT THE MODE BYTE
00003B86 89D0                <1>       mov    eax, edx            ; Buffer address
00003B88 E604                <1>       OUT    DMA+4,AL            ; OUTPUT LOW ADDRESS
                             <1>       IODELAY                           ; WAIT FOR I/O
00003B8A EB00                <2>  jmp short $+2
00003B8C EB00                <2>  jmp short $+2
00003B8E 88E0                <1>       MOV    AL,AH
00003B90 E604                <1>       OUT    DMA+4,AL            ; OUTPUT HIGH ADDRESS
00003B92 C1E810              <1>       shr    eax, 16
                             <1>       IODELAY                           ; I/O WAIT STATE
00003B95 EB00                <2>  jmp short $+2
00003B97 EB00                <2>  jmp short $+2
00003B99 E681                <1>       OUT    081H,AL                    ; OUTPUT highest BITS TO PAGE REGISTER
                             <1>       IODELAY
00003B9B EB00                <2>  jmp short $+2
00003B9D EB00                <2>  jmp short $+2
00003B9F 6689C8              <1>       mov    ax, cx              ; Byte count - 1
00003BA2 E605                <1>       OUT    DMA+5,AL            ; LOW BYTE OF COUNT
                             <1>       IODELAY                           ; WAIT FOR I/O
00003BA4 EB00                <2>  jmp short $+2
00003BA6 EB00                <2>  jmp short $+2
00003BA8 88E0                <1>       MOV    AL, AH
00003BAA E605                <1>       OUT    DMA+5,AL            ; HIGH BYTE OF COUNT
                             <1>       IODELAY
00003BAC EB00                <2>  jmp short $+2
00003BAE EB00                <2>  jmp short $+2
00003BB0 FB                  <1>       STI                        ; RE-ENABLE INTERRUPTS
00003BB1 B002                <1>       MOV    AL, 2               ; MODE FOR 8237
00003BB3 E60A                <1>       OUT    DMA+10, AL          ; INITIALIZE THE DISKETTE CHANNEL
                             <1>
00003BB5 F8                  <1>       clc    ; 04/02/2016
00003BB6 C3                  <1>       retn
                             <1>
                             <1> dma_bnd_err_stc:
00003BB7 F9                  <1>       stc
                             <1> dma_bnd_err:
00003BB8 C605[C0580100]09    <1>       MOV    byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
00003BBF C3                  <1>       RETn                       ; CY SET BY ABOVE IF ERROR
                             <1>
                             <1> ;; 16/12/2014
                             <1> ;;    CLI                        ; DISABLE INTERRUPTS DURING DMA SET-UP
                             <1> ;;    OUT    DMA+12,AL           ; SET THE FIRST/LA5T F/F
                             <1> ;;    ;JMP   $+2                 ; WAIT FOR I/O
                             <1> ;;    IODELAY
                             <1> ;;    OUT    DMA+11,AL           ; OUTPUT THE MODE BYTE
                             <1> ;;    ;SIODELAY
                             <1> ;;      ;CMP AL, 42H                     ; DMA VERIFY COMMAND
                             <1> ;;      ;JNE short NOT_VERF              ; NO
                             <1> ;;      ;XOR AX, AX              ; START ADDRESS
                             <1> ;;      ;JMP SHORT J33
                             <1> ;;;NOT_VERF:
                             <1> ;;    ;MOV   AX,ES               ; GET THE ES VALUE
                             <1> ;;    ;ROL   AX,4                ; ROTATE LEFT
                             <1> ;;    ;MOV   CH,AL               ; GET HIGHEST NIBBLE OF ES TO CH
                             <1> ;;    ;AND   AL,11110000B        ; ZERO THE LOW NIBBLE FROM SEGMENT
                             <1> ;;    ;ADD   AX,[BP+2]           ; TEST FOR CARRY FROM ADDITION
                             <1> ;;    mov    eax, [ebp+4] ; 06/02/2015
                             <1> ;;    ;JNC   short J33
                             <1> ;;    ;INC   CH                  ; CARRY MEANS HIGH 4 BITS MUST BE INC
                             <1> ;;;J33:
                             <1> ;;    PUSH   eAX                 ; SAVE START ADDRESS
                             <1> ;;    OUT    DMA+4,AL            ; OUTPUT LOW ADDRESS
                             <1> ;;    ;JMP   $+2                 ; WAIT FOR I/O
                             <1> ;;    IODELAY
                             <1> ;;    MOV    AL,AH
                             <1> ;;    OUT    DMA+4,AL            ; OUTPUT HIGH ADDRESS
                             <1> ;;    shr    eax, 16           ; 07/02/2015
                             <1> ;;    ;MOV   AL,CH               ; GET HIGH 4 BITS
                             <1> ;;    ;JMP   $+2                 ; I/O WAIT STATE
                             <1> ;;    IODELAY
                             <1> ;;    ;AND   AL,00001111B
                             <1> ;;    OUT    081H,AL                    ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
                             <1> ;;    ;SIODELAY
                             <1> ;;
                             <1> ;;;----- DETERMINE COUNT
                             <1> ;;    sub    eax, eax ; 08/02/2015
                             <1> ;;    MOV    AX, SI              ; AL =  # OF SECTORS
                             <1> ;;    XCHG   AL, AH              ; AH =  # OF SECTORS
                             <1> ;;    SUB    AL, AL              ; AL = 0, AX = # SECTORS * 256
                             <1> ;;    SHR    AX, 1               ; AX = # SECTORS * 128
                             <1> ;;    PUSH   AX                  ; SAVE # OF SECTORS * 128
                             <1> ;;    MOV    DL, 3               ; GET BYTES/SECTOR PARAMETER
                             <1> ;;    CALL   GET_PARM            ; "
                             <1> ;;    MOV    CL,AH               ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
                             <1> ;;    POP   AX                  ; AX = # SECTORS * 128
                             <1> ;;    SHL   AX,CL               ; SHIFT BY PARAMETER VALUE
                             <1> ;;    DEC   AX                  ; -1 FOR DMA VALUE
                             <1> ;;    PUSH  eAX  ; 08/02/2015   ; SAVE COUNT VALUE
                             <1> ;;    OUT   DMA+5,AL            ; LOW BYTE OF COUNT
                             <1> ;;    ;JMP  $+2                 ; WAIT FOR I/O
                             <1> ;;    IODELAY
                             <1> ;;    MOV   AL, AH
                             <1> ;;    OUT   DMA+5,AL            ; HIGH BYTE OF COUNT
                             <1> ;;    ;IODELAY
                             <1> ;;    STI                       ; RE-ENABLE INTERRUPTS
                             <1> ;;    POP   eCX  ; 08/02/2015   ; RECOVER COUNT VALUE
                             <1> ;;    POP   eAX  ; 08/02/2015   ; RECOVER ADDRESS VALUE
                             <1> ;;    ;ADD  AX, CX              ; ADD, TEST FOR 64K OVERFLOW
                             <1> ;;    add   ecx, eax ; 08/02/2015
                             <1> ;;    MOV   AL, 2               ; MODE FOR 8237
                             <1> ;;    ;JMP  $+2                 ; WAIT FOR I/O
                             <1> ;;    SIODELAY
                             <1> ;;    OUT   DMA+10, AL          ; INITIALIZE THE DISKETTE CHANNEL
                             <1> ;;    ;JNC  short NO_BAD        ; CHECK FOR ERROR
                             <1> ;;    jc    short dma_bnd_err ; 08/02/2015
                             <1> ;;    and   ecx, 0FFF00000h ; 16 MB limit
                             <1> ;;    jz    short NO_BAD
                             <1> ;;dma_bnd_err:
                             <1> ;;    MOV   byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
                             <1> ;;NO_BAD:
                             <1> ;;    RETn                      ; CY SET BY ABOVE IF ERROR
                             <1>
                             <1> ;------------------------------------------------------------------------------
                             <1> ; FMTDMA_SET
                             <1> ;     THIS ROUTINE SETS UP THE DMA CONTROLLER FOR A FORMAT OPERATION.
                             <1> ;
                             <1> ; ON ENTRY:  NOTHING REQUIRED
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;------------------------------------------------------------------------------
                             <1>
                             <1> FMTDMA_SET:
                             <1> ;; 20/02/2015 modification
00003BC0 8B5504              <1>       mov   edx, [ebp+4]        ; Buffer address
00003BC3 F7C20000F0FF        <1>       test  edx, 0FFF00000h           ; 16 MB limit
00003BC9 75EC                <1>       jnz   short dma_bnd_err_stc
                             <1>       ;
00003BCB 6652                <1>       push  dx                  ; *
00003BCD B204                <1>       mov   DL, 4               ; SECTORS/TRACK VALUE IN PARM TABLE
00003BCF E8D0020000          <1>       call  GET_PARM            ; "
00003BD4 88E0                <1>       mov   al, ah              ; AL = SECTORS/TRACK VALUE
00003BD6 28E4                <1>       sub   ah, ah              ; AX = SECTORS/TRACK VALUE
00003BD8 66C1E002            <1>       shl   ax, 2               ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
00003BDC 6648                <1>       dec   ax                  ; -1 FOR DMA VALUE
00003BDE 6689C1              <1>       mov   cx, ax
00003BE1 665A                <1>       pop   dx                  ; *
00003BE3 6601CA              <1>       add   dx, cx              ; check for overflow
00003BE6 72D0                <1>       jc    short dma_bnd_err
                             <1>       ;
00003BE8 6629CA              <1>       sub   dx, cx              ; Restore start address
                             <1>       ;
00003BEB B04A                <1>       MOV   AL, 04AH            ; WILL WRITE TO THE DISKETTE
00003BED FA                  <1>       CLI                       ; DISABLE INTERRUPTS DURING DMA SET-UP
00003BEE E60C                <1>       OUT   DMA+12,AL           ; SET THE FIRST/LA5T F/F
                             <1>       IODELAY                         ; WAIT FOR I/O
00003BF0 EB00                <2>  jmp short $+2
00003BF2 EB00                <2>  jmp short $+2
00003BF4 E60B                <1>       OUT   DMA+11,AL           ; OUTPUT THE MODE BYTE
00003BF6 89D0                <1>       mov   eax, edx            ; Buffer address
00003BF8 E604                <1>       OUT   DMA+4,AL            ; OUTPUT LOW ADDRESS
                             <1>       IODELAY                         ; WAIT FOR I/O
00003BFA EB00                <2>  jmp short $+2
00003BFC EB00                <2>  jmp short $+2
00003BFE 88E0                <1>       MOV   AL,AH
00003C00 E604                <1>       OUT   DMA+4,AL            ; OUTPUT HIGH ADDRESS
00003C02 C1E810              <1>       shr   eax, 16
                             <1>       IODELAY                         ; I/O WAIT STATE
00003C05 EB00                <2>  jmp short $+2
00003C07 EB00                <2>  jmp short $+2
00003C09 E681                <1>       OUT   081H,AL                   ; OUTPUT highest BITS TO PAGE REGISTER
                             <1>       IODELAY
00003C0B EB00                <2>  jmp short $+2
00003C0D EB00                <2>  jmp short $+2
00003C0F 6689C8              <1>       mov   ax, cx              ; Byte count - 1
00003C12 E605                <1>       OUT   DMA+5,AL            ; LOW BYTE OF COUNT
                             <1>       IODELAY                         ; WAIT FOR I/O
00003C14 EB00                <2>  jmp short $+2
00003C16 EB00                <2>  jmp short $+2
00003C18 88E0                <1>       MOV   AL, AH
00003C1A E605                <1>       OUT   DMA+5,AL            ; HIGH BYTE OF COUNT
                             <1>       IODELAY
00003C1C EB00                <2>  jmp short $+2
00003C1E EB00                <2>  jmp short $+2
00003C20 FB                  <1>       STI                       ; RE-ENABLE INTERRUPTS
00003C21 B002                <1>       MOV   AL, 2               ; MODE FOR 8237
00003C23 E60A                <1>       OUT   DMA+10, AL          ; INITIALIZE THE DISKETTE CHANNEL
00003C25 C3                  <1>       retn
                             <1>
                             <1> ;; 08/02/2015 - Protected Mode Modification
                             <1> ;;    MOV   AL, 04AH            ; WILL WRITE TO THE DISKETTE
                             <1> ;;    CLI                       ; DISABLE INTERRUPTS DURING DMA SET-UP
                             <1> ;;    OUT   DMA+12,AL           ; SET THE FIRST/LA5T F/F
                             <1> ;;    ;JMP  $+2                 ; WAIT FOR I/O
                             <1> ;;    IODELAY
                             <1> ;;    OUT   DMA+11,AL           ; OUTPUT THE MODE BYTE
                             <1> ;;    ;MOV  AX,ES               ; GET THE ES VALUE
                             <1> ;;    ;ROL  AX,4                ; ROTATE LEFT
                             <1> ;;    ;MOV  CH,AL               ; GET HIGHEST NIBBLE OF ES TO CH
                             <1> ;;    ;AND  AL,11110000B        ; ZERO THE LOW NIBBLE FROM SEGMENT
                             <1> ;;    ;ADD  AX,[BP+2]           ; TEST FOR CARRY FROM ADDITION
                             <1> ;;    ;JNC  short J33A
                             <1> ;;    ;INC  CH                  ; CARRY MEANS HIGH 4 BITS MUST BE INC
                             <1> ;;    mov   eax, [ebp+4] ; 08/02/2015
                             <1> ;;;J33A:
                             <1> ;;    PUSH  eAX ; 08/02/2015    ; SAVE START ADDRESS
                             <1> ;;    OUT   DMA+4,AL            ; OUTPUT LOW ADDRESS
                             <1> ;;    ;JMP  $+2                 ; WAIT FOR I/O
                             <1> ;;    IODELAY
                             <1> ;;    MOV   AL,AH
                             <1> ;;    OUT   DMA+4,AL            ; OUTPUT HIGH ADDRESS
                             <1> ;;    shr   eax, 16 ; 08/02/2015
                             <1> ;;    ;MOV  AL,CH               ; GET HIGH 4 BITS
                             <1> ;;    ;JMP  $+2                 ; I/O WAIT STATE
                             <1> ;;    IODELAY
                             <1> ;;    ;AND  AL,00001111B
                             <1> ;;    OUT   081H,AL                   ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
                             <1> ;;
                             <1> ;;;----- DETERMINE COUNT
                             <1> ;;    sub   eax, eax ; 08/02/2015
                             <1> ;;    MOV   DL, 4               ; SECTORS/TRACK VALUE IN PARM TABLE
                             <1> ;;    CALL  GET_PARM            ; "
                             <1> ;;    XCHG  AL, AH              ; AL = SECTORS/TRACK VALUE
                             <1> ;;    SUB   AH, AH              ; AX = SECTORS/TRACK VALUE
                             <1> ;;    SHL   AX, 2               ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
                             <1> ;;    DEC   AX                  ; -1 FOR DMA VALUE
                             <1> ;;    PUSH  eAX    ; 08/02/2015 ; SAVE # OF BYTES TO BE TRANSFERED
                             <1> ;;    OUT   DMA+5,AL            ; LOW BYTE OF COUNT
                             <1> ;;    ;JMP  $+2                 ; WAIT FOR I/O
                             <1> ;;    IODELAY
                             <1> ;;    MOV   AL, AH
                             <1> ;;    OUT   DMA+5,AL            ; HIGH BYTE OF COUNT
                             <1> ;;    STI                       ; RE-ENABLE INTERRUPTS
                             <1> ;;    POP   eCX    ; 08/02/2015 ; RECOVER COUNT VALUE
                             <1> ;;    POP   eAX    ; 08/02/2015 ; RECOVER ADDRESS VALUE
                             <1> ;;    ;ADD  AX, CX              ; ADD, TEST FOR 64K OVERFLOW
                             <1> ;;    add   ecx, eax ; 08/02/2015
                             <1> ;;    MOV   AL, 2               ; MODE FOR 8237
                             <1> ;;    ;JMP  $+2                 ; WAIT FOR I/O
                             <1> ;;    SIODELAY
                             <1> ;;    OUT   DMA+10, AL          ; INITIALIZE THE DISKETTE CHANNEL
                             <1> ;;    ;JNC  short FMTDMA_OK           ; CHECK FOR ERROR
                             <1> ;;    jc    short fmtdma_bnd_err ; 08/02/2015
                             <1> ;;    and   ecx, 0FFF00000h  ; 16 MB limit
                             <1> ;;    jz    short FMTDMA_OK
                             <1> ;;    stc   ; 20/02/2015
                             <1> ;;fmtdma_bnd_err:
                             <1> ;;    MOV   byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
                             <1> ;;FMTDMA_OK:
                             <1> ;;    RETn                      ; CY SET BY ABOVE IF ERROR
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; NEC_INIT
                             <1> ;     THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND INITIALIZES
                             <1> ;     THE NEC FOR THE READ/WRITE/VERIFY/FORMAT OPERATION.
                             <1> ;
                             <1> ; ON ENTRY:  AH = NEC COMMAND TO BE PERFORMED
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;-------------------------------------------------------------------------------
                             <1> NEC_INIT:
00003C26 6650                <1>       PUSH  AX                  ; SAVE NEC COMMAND
00003C28 E8BC020000          <1>       CALL  MOTOR_ON            ; TURN MOTOR ON FOR SPECIFIC DRIVE
                             <1>
                             <1> ;-----       DO THE SEEK OPERATION
                             <1>
00003C2D 8A6D01              <1>       MOV   CH,[eBP+1]          ; CH = TRACK #
00003C30 E8AF030000          <1>       CALL  SEEK                ; MOVE TO CORRECT TRACK
00003C35 6658                <1>       POP   AX                  ; RECOVER COMMAND
00003C37 721E                <1>       JC    short ER_1          ; ERROR ON SEEK
00003C39 BB[573C0000]        <1>       MOV   eBX, ER_1           ; LOAD ERROR ADDRESS
00003C3E 53                  <1>       PUSH  eBX                 ; PUSH NEC_OUT ERROR RETURN
                             <1>
                             <1> ;-----       SEND OUT THE PARAMETERS TO THE CONTROLLER
                             <1>
00003C3F E866030000          <1>       CALL  NEC_OUTPUT          ; OUTPUT THE OPERATION COMMAND
00003C44 6689F0              <1>       MOV   AX,SI               ; AH = HEAD #
00003C47 89FB                <1>       MOV   eBX,eDI                   ; BL = DRIVE #
00003C49 C0E402              <1>       SAL   AH,2                ; MOVE IT TO BIT 2
00003C4C 80E404              <1>       AND   AH,00000100B        ; ISOLATE THAT BIT
00003C4F 08DC                <1>       OR    AH,BL               ; OR IN THE DRIVE NUMBER
00003C51 E854030000          <1>       CALL  NEC_OUTPUT          ; FALL THRU CY SET IF ERROR
00003C56 5B                  <1>       POP   eBX                 ; THROW AWAY ERROR RETURN
                             <1> ER_1:
00003C57 C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; RWV_COM
                             <1> ;     THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC TO THE
                             <1> ;     READ/WRITE/VERIFY OPERATIONS.
                             <1> ;
                             <1> ; ON ENTRY:  CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;-------------------------------------------------------------------------------
                             <1> RWV_COM:
00003C58 B8[A33C0000]        <1>       MOV   eAX, ER_2           ; LOAD ERROR ADDRESS
00003C5D 50                  <1>       PUSH  eAX                 ; PUSH NEC_OUT ERROR RETURN
00003C5E 8A6501              <1>       MOV   AH,[eBP+1]          ; OUTPUT TRACK #
00003C61 E844030000          <1>       CALL  NEC_OUTPUT
00003C66 6689F0              <1>       MOV   AX,SI               ; OUTPUT HEAD #
00003C69 E83C030000          <1>       CALL  NEC_OUTPUT
```

```
00003C6E 8A6500              <1>         MOV     AH,[eBP]                ; OUTPUT SECTOR #
00003C71 E834030000          <1>       CALL   NEC_OUTPUT
00003C76 B203                <1>       MOV    DL,3                ; BYTES/SECTOR PARAMETER FROM BLOCK
00003C78 E827020000          <1>       CALL   GET_PARM            ; ... TO THE NEC
00003C7D E828030000          <1>       CALL   NEC_OUTPUT          ; OUTPUT TO CONTROLLER
00003C82 B204                <1>       MOV    DL,4                ; EOT PARAMETER FROM BLOCK
00003C84 E81B020000          <1>       CALL   GET_PARM            ; ... TO THE NEC
00003C89 E81C030000          <1>       CALL   NEC_OUTPUT          ; OUTPUT TO CONTROLLER
00003C8E 8A6305              <1>         MOV     AH, [eBX+MD.GAP]        ; GET GAP LENGTH
                             <1> _R15:
00003C91 E814030000          <1>       CALL   NEC_OUTPUT
00003C96 B206                <1>       MOV    DL,6                ; DTL PARAMETER PROM BLOCK
00003C98 E807020000          <1>       CALL   GET_PARM            ;  TO THE NEC
00003C9D E808030000          <1>       CALL   NEC_OUTPUT          ; OUTPUT TO CONTROLLER
00003CA2 58                  <1>       POP    eAX                 ; THROW AWAY ERROR EXIT
                             <1> ER_2:
00003CA3 C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; NEC_TERM
                             <1> ;     THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS THE STATUS
                             <1> ;     FROM THE NEC FOR THE READ/WRITE/VERIFY/FORWAT OPERATION.
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;-------------------------------------------------------------------------------
                             <1> NEC_TERM:
                             <1>
                             <1> ;-----      LET THE OPERATION HAPPEN
                             <1>
00003CA4 56                  <1>       PUSH   eSI                 ; SAVE HEAD #, # OF SECTORS
00003CA5 E80D040000          <1>       CALL   WAIT_INT            ; WAIT FOR THE INTERRUPT
00003CAA 9C                  <1>       PUSHF
00003CAB E837040000          <1>       CALL   RESULTS                    ; GET THE NEC STATUS
00003CB0 724B                <1>       JC     short SET_END_POP
00003CB2 9D                  <1>       POPF
00003CB3 723E                <1>       JC     short SET_END       ; LOOK FOR ERROR
                             <1>
                             <1> ;-----      CHECK THE RESULTS RETURNED BY THE CONTROLLER
                             <1>
00003CB5 FC                  <1>       CLD                        ; SET THE CORRECT DIRECTION
00003CB6 BE[C1580100]        <1>       MOV    eSI, NEC_STATUS            ; POINT TO STATUS FIELD
00003CBB AC                  <1>       lodsb                      ; GET ST0
00003CBC 24C0                <1>       AND    AL,11000000B        ; TEST FOR NORMAL TERMINATION
00003CBE 7433                <1>       JZ     short SET_END
00003CC0 3C40                <1>       CMP    AL,01000000B        ; TEST FOR ABNORMAL TERMINATION
00003CC2 7527                <1>       JNZ    short J18           ; NOT ABNORMAL, BAD NEC
                             <1>
                             <1> ;-----      ABNORMAL TERMINATION, FIND OUT WHY
                             <1>
00003CC4 AC                  <1>       lodsb                      ; GET ST1
00003CC5 D0E0                <1>       SAL    AL,1                ; TEST FOR EDT FOUND
00003CC7 B404                <1>       MOV    AH,RECORD_NOT_FND
00003CC9 7222                <1>       JC     short J19
00003CCB C0E002              <1>       SAL    AL,2
00003CCE B410                <1>       MOV    AH,BAD_CRC
00003CD0 721B                <1>       JC     short J19
00003CD2 D0E0                <1>       SAL    AL,1                ; TEST FOR DMA OVERRUN
00003CD4 B408                <1>       MOV    AH,BAD_DMA
00003CD6 7215                <1>       JC     short J19
00003CD8 C0E002              <1>       SAL    AL,2                ; TEST FOR RECORD NOT FOUND
00003CDB B404                <1>       MOV    AH,RECORD_NOT_FND
00003CDD 720E                <1>       JC     short J19
00003CDF D0E0                <1>       SAL    AL,1
00003CE1 B403                <1>       MOV    AH,WRITE_PROTECT    ; TEST FOR WRITE_PROTECT
00003CE3 7208                <1>       JC     short J19
00003CE5 D0E0                <1>       SAL    AL,1                ; TEST MISSING ADDRESS MARK
00003CE7 B402                <1>       MOV    AH,BAD_ADDR_MARK
00003CE9 7202                <1>       JC     short J19
                             <1>
                             <1> ;-----      NEC MUST HAVE FAILED
                             <1> J18:
00003CEB B420                <1>       MOV    AH,BAD_NEC
                             <1> J19:
00003CED 0825[C0580100]      <1>       OR     [DSKETTE_STATUS], AH
                             <1> SET_END:
00003CF3 803D[C0580100]01    <1>       CMP    byte [DSKETTE_STATUS], 1 ; SET ERROR CONDITION
00003CFA F5                  <1>       CMC
00003CFB 5E                  <1>       POP    eSI
00003CFC C3                  <1>       RETn                       ; RESTORE HEAD #, # OF SECTORS
                             <1>
                             <1> SET_END_POP:
00003CFD 9D                  <1>       POPF
00003CFE EBF3                <1>       JMP    SHORT SET_END
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; DSTATE:    ESTABLISH STATE UPON SUCCESSFUL OPERATION.
                             <1> ;-------------------------------------------------------------------------------
                             <1> DSTATE:
00003D00 803D[C0580100]00    <1>       CMP    byte [DSKETTE_STATUS],0    ; CHECK FOR ERROR
00003D07 753E                <1>       JNZ    short SETBAC            ; IF ERROR JUMP
00003D09 808F[CD580100]10    <1>       OR     byte [DSK_STATE+eDI],MED_DET ; NO ERROR, MARK MEDIA AS DETERMINED
00003D10 F687[CD580100]04    <1>       TEST   byte [DSK_STATE+eDI],DRV_DET ; DRIVE DETERMINED ?
00003D17 752E                <1>       JNZ    short SETBAC        ; IF DETERMINED NO TRY TO DETERMINE
00003D19 8A87[CD580100]      <1>       MOV    AL,[DSK_STATE+eDI]  ; LOAD STATE
00003D1F 24C0                <1>       AND    AL,RATE_MSK         ; KEEP ONLY RATE
00003D21 3C80                <1>       CMP    AL,RATE_250         ; RATE 250 ?
00003D23 751B                <1>       JNE    short M_12          ; NO, MUST BE 1.2M OR 1.44M DRIVE
                             <1>
                             <1> ;-----      CHECK IF IT IS 1.44M
                             <1>
00003D25 E871010000          <1>       CALL   CMOS_TYPE           ; RETURN DRIVE TYPE IN (AL)
                             <1>       ;;20/02/2015
                             <1>       ;;JC   short M_12          ; CMOS BAD
```

```
00003D2A 7414                <1>       jz     short M_12 ;; 20/02/2015
00003D2C 3C04                <1>       CMP    AL, 4               ; 1.44MB DRIVE ?
00003D2E 7410                <1>       JE     short M_12          ; YES
                             <1> M_720:
00003D30 80A7[CD580100]FD    <1>       AND    byte [DSK_STATE+eDI], ~FMT_CAPA ; TURN OFF FORMAT CAPABILITY
00003D37 808F[CD580100]04    <1>       OR     byte [DSK_STATE+eDI],DRV_DET  ; MARK DRIVE DETERMINED
00003D3E EB07                <1>       JMP    SHORT SETBAC        ; BACK
                             <1> M_12:
00003D40 808F[CD580100]06    <1>       OR     byte [DSK_STATE+eDI],DRV_DET+FMT_CAPA
                             <1>                                  ; TURN ON DETERMINED & FMT CAPA
                             <1> SETBAC:
00003D47 C3                  <1>       RETn
                             <1>
                             <1> ;------------------------------------------------------------------------------
                             <1> ; RETRY
                             <1> ;     DETERMINES WHETHER A RETRY IS NECESSARY.
                             <1> ;     IF RETRY IS REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.
                             <1> ;
                             <1> ; ON EXIT:   CY = 1 FOR RETRY, CY = 0 FOR NO RETRY
                             <1> ;------------------------------------------------------------------------------
                             <1> RETRY:
00003D48 803D[C0580100]00    <1>       CMP    byte [DSKETTE_STATUS],0    ; GET STATUS OF OPERATION
00003D4F 7445                <1>       JZ     short NO_RETRY             ; SUCCESSFUL OPERATION
00003D51 803D[C0580100]80    <1>       CMP    byte [DSKETTE_STATUS],TIME_OUT ; IF TIME OUT NO RETRY
00003D58 743C                <1>       JZ     short NO_RETRY
00003D5A 8AA7[CD580100]      <1>       MOV    AH,[DSK_STATE+eDI]  ; GET MEDIA STATE OF DRIVE
00003D60 F6C410              <1>       TEST   AH,MED_DET          ; ESTABLISHED/DETERMINED ?
00003D63 7531                <1>       JNZ    short NO_RETRY             ; IF ESTABLISHED STATE THEN TRUE ERROR
00003D65 80E4C0              <1>       AND    AH,RATE_MSK         ; ISOLATE RATE
00003D68 8A2D[C8580100]      <1>       MOV    CH,[LASTRATE]       ; GET START OPERATION STATE
00003D6E C0C504              <1>       ROL    CH,4                ; TO CORRESPONDING BITS
00003D71 80E5C0              <1>       AND    CH,RATE_MSK         ; ISOLATE RATE BITS
00003D74 38E5                <1>       CMP    CH,AH               ; ALL RATES TRIED
00003D76 741E                <1>       JE     short NO_RETRY             ; IF YES, THEN TRUE ERROR
                             <1>
                             <1> ;     SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
                             <1> ;      00000000B (500) -> 10000000B    (250)
                             <1> ;      10000000B (250) -> 01000000B    (300)
                             <1> ;      01000000B (300) -> 00000000B    (500)
                             <1>
00003D78 80FC01              <1>       CMP    AH,RATE_500+1       ; SET CY FOR RATE 500
00003D7B D0DC                <1>       RCR    AH,1                ; TO NEXT STATE
00003D7D 80E4C0              <1>       AND    AH,RATE_MSK         ; KEEP ONLY RATE BITS
00003D80 80A7[CD580100]1F    <1>       AND    byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP)
                             <1>                                  ; RATE, DBL STEP OFF
00003D87 08A7[CD580100]      <1>       OR     [DSK_STATE+eDI],AH  ; TURN ON NEW RATE
00003D8D C605[C0580100]00    <1>       MOV    byte [DSKETTE_STATUS],0    ; RESET STATUS FOR RETRY
00003D94 F9                  <1>       STC                        ; SET CARRY FOR RETRY
00003D95 C3                  <1>       RETn                       ; RETRY RETURN
                             <1>
                             <1> NO_RETRY:
00003D96 F8                  <1>       CLC                        ; CLEAR CARRY NO RETRY
00003D97 C3                  <1>       RETn                       ; NO RETRY RETURN
                             <1>
                             <1> ;------------------------------------------------------------------------------
                             <1> ; NUM_TRANS
                             <1> ;     THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT WERE
                             <1> ;     ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
                             <1> ;
                             <1> ; ON ENTRY:  [BP+1] = TRACK
                             <1> ;            SI-HI  = HEAD
                             <1> ;            [BP]   = START SECTOR
                             <1> ;
                             <1> ; ON EXIT:   AL = NUMBER ACTUALLY TRANSFERRED
                             <1> ;------------------------------------------------------------------------------
                             <1> NUM_TRANS:
00003D98 30C0                <1>       XOR    AL,AL               ; CLEAR FOR ERROR
00003D9A 803D[C0580100]00    <1>       CMP    byte [DSKETTE_STATUS],0    ; CHECK FOR ERROR
00003DA1 752C                <1>       JNZ    NT_OUT              ; IF ERROR 0 TRANSFERRED
00003DA3 B204                <1>       MOV    DL,4                ; SECTORS/TRACK OFFSET TO DL
00003DA5 E8FA000000          <1>       CALL   GET_PARM            ; AH = SECTORS/TRACK
00003DAA 8A1D[C6580100]      <1>       MOV    BL, [NEC_STATUS+5]  ; GET ENDING SECTOR
00003DB0 6689F1              <1>       MOV    CX,SI               ; CH = HEAD # STARTED
00003DB3 3A2D[C5580100]      <1>       CMP    CH, [NEC_STATUS+4]  ; GET HEAD ENDED UP ON
00003DB9 750D                <1>       JNZ    DIF_HD              ; IF ON SAME HEAD, THEN NO ADJUST
00003DBB 8A2D[C4580100]      <1>       MOV    CH, [NEC_STATUS+3]  ; GET TRACK ENDED UP ON
00003DC1 3A6D01              <1>       CMP    CH,[eBP+1]          ; IS IT ASKED FOR TRACK
00003DC4 7404                <1>       JZ     short SAME_TRK             ; IF SAME TRACK NO INCREASE
00003DC6 00E3                <1>       ADD    BL,AH               ; ADD SECTORS/TRACK
                             <1> DIF_HD:
00003DC8 00E3                <1>       ADD    BL,AH               ; ADD SECTORS/TRACK
                             <1> SAME_TRK:
00003DCA 2A5D00              <1>       SUB    BL,[eBP]            ; SUBTRACT START FROM END
00003DCD 88D8                <1>       MOV    AL,BL               ; TO AL
                             <1> NT_OUT:
00003DCF C3                  <1>       RETn
                             <1>
                             <1> ;------------------------------------------------------------------------------
                             <1> ; SETUP_END
                             <1> ;     RESTORES @MOTOR_COUNT TO PARAMETER PROVIDED IN TABLE
                             <1> ;     AND LOADS @DSKETTE_STATUS TO AH, AND SETS CY.
                             <1> ;
                             <1> ; ON EXIT:
                             <1> ;     AH, @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;------------------------------------------------------------------------------
                             <1> SETUP_END:
00003DD0 B202                <1>       MOV    DL,2                ; GET THE MOTOR WAIT PARAMETER
00003DD2 6650                <1>       PUSH   AX                  ; SAVE NUMBER TRANSFERRED
00003DD4 E8CB000000          <1>       CALL   GET_PARM
00003DD9 8825[BF580100]      <1>       MOV    [MOTOR_COUNT],AH    ; STORE UPON RETURN
00003DDF 6658                <1>       POP    AX                  ; RESTORE NUMBER TRANSFERRED
00003DE1 8A25[C0580100]      <1>       MOV    AH, [DSKETTE_STATUS]       ; GET STATUS OF OPERATION
00003DE7 08E4                <1>       OR     AH,AH               ; CHECK FOR ERROR
```

```
00003DE9 7402                <1>       JZ    short NUN_ERR       ; NO ERROR
00003DEB 30C0                <1>       XOR   AL,AL               ; CLEAR NUMBER RETURNED
                             <1> NUN_ERR:
00003DED 80FC01              <1>       CMP   AH,1                ; SET THE CARRY FLAG TO INDICATE
00003DF0 F5                  <1>       CMC                       ; SUCCESS OR FAILURE
00003DF1 C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; SETUP_DBL
                             <1> ;     CHECK DOUBLE STEP.
                             <1> ;
                             <1> ; ON ENTRY : DI = DRIVE
                             <1> ;
                             <1> ; ON EXIT :  CY = 1 MEANS ERROR
                             <1> ;-------------------------------------------------------------------------------
                             <1> SETUP_DBL:
00003DF2 8AA7[CD580100]      <1>       MOV   AH, [DSK_STATE+eDI] ; ACCESS STATE
00003DF8 F6C410              <1>       TEST  AH,MED_DET          ; ESTABLISHED STATE ?
00003DFB 757E                <1>       JNZ   short NO_DBL               ; IF ESTABLISHED THEN DOUBLE DONE
                             <1>
                             <1> ;-----      CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE
                             <1>
00003DFD C605[BD580100]00    <1>       MOV   byte [SEEK_STATUS],0       ; SET RECALIBRATE REQUIRED ON ALL DRIVES
00003E04 E8E0000000          <1>       CALL  MOTOR_ON            ; ENSURE MOTOR STAY ON
00003E09 B500                <1>       MOV   CH,0                ; LOAD TRACK 0
00003E0B E8D4010000          <1>       CALL  SEEK                ; SEEK TO TRACK 0
00003E10 E868000000          <1>       CALL  READ_ID                    ; READ ID FUNCTION
00003E15 7249                <1>       JC    short SD_ERR        ; IF ERROR NO TRACK 0
                             <1>
                             <1> ;-----      INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)
                             <1>
00003E17 66B95004            <1>       MOV   CX,0450H            ; START, MAX TRACKS
00003E1B F687[CD580100]01    <1>       TEST  byte [DSK_STATE+eDI],TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
00003E22 7402                <1>       JZ    short CNT_OK        ; IF NOT COUNT IS SETUP
00003E24 B1A0                <1>       MOV   CL,0A0H                    ; MAXIMUM TRACK 1.2 MB
                             <1>
                             <1> ;     ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,
                             <1> ;     MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT
                             <1> ;     THEN SET DOUBLE STEP ON.
                             <1>
                             <1> CNT_OK:
00003E26 C605[BF580100]FF    <1>         MOV     byte [MOTOR_COUNT], 0FFH ; ENSURE MOTOR STAYS ON FOR OPERATION
00003E2D 6651                <1>       PUSH  CX                  ; SAVE TRACK, COUNT
00003E2F C605[C0580100]00    <1>       MOV   byte [DSKETTE_STATUS],0   ; CLEAR STATUS, EXPECT ERRORS
00003E36 6631C0              <1>       XOR   AX,AX               ; CLEAR AX
00003E39 D0ED                <1>       SHR   CH,1                ; HALVE TRACK, CY = HEAD
00003E3B C0D003              <1>       RCL   AL,3                ; AX = HEAD IN CORRECT BIT
00003E3E 6650                <1>       PUSH  AX                  ; SAVE HEAD
00003E40 E89F010000          <1>       CALL  SEEK                ; SEEK TO TRACK
00003E45 6658                <1>       POP   AX                  ; RESTORE HEAD
00003E47 6609C7              <1>       OR    DI,AX               ; DI = HEAD OR'ED DRIVE
00003E4A E82E000000          <1>       CALL  READ_ID                    ; READ ID HEAD 0
00003E4F 9C                  <1>       PUSHF                     ; SAVE RETURN FROM READ_ID
00003E50 6681E7FB00          <1>       AND   DI,11111011B        ; TURN OFF HEAD 1 BIT
00003E55 9D                  <1>       POPF                      ; RESTORE ERROR RETURN
00003E56 6659                <1>       POP   CX                  ; RESTORE COUNT
00003E58 7308                <1>       JNC   short DO_CHK        ; IF OK, ASKED = RETURNED TRACK ?
00003E5A FEC5                <1>       INC   CH                  ; INC FOR NEXT TRACK
00003E5C 38CD                <1>       CMP   CH,CL               ; REACHED MAXIMUM YET
00003E5E 75C6                <1>       JNZ   short CNT_OK        ; CONTINUE TILL ALL TRIED
                             <1>
                             <1> ;-----      FALL THRU, READ ID FAILED FOR ALL TRACKS
                             <1>
                             <1> SD_ERR:
00003E60 F9                  <1>       STC                       ; SET CARRY FOR ERROR
00003E61 C3                  <1>       RETn                      ; SETUP_DBL ERROR EXIT
                             <1>
                             <1> DO_CHK:
00003E62 8A0D[C4580100]      <1>       MOV   CL, [NEC_STATUS+3]  ; LOAD RETURNED TRACK
00003E68 888F[D1580100]      <1>       MOV   [DSK_TRK+eDI], CL   ; STORE TRACK NUMBER
00003E6E D0ED                <1>       SHR   CH,1                ; HALVE TRACK
00003E70 38CD                <1>       CMP   CH,CL               ; IS IT THE SAME AS ASKED FOR TRACK
00003E72 7407                <1>       JZ    short NO_DBL        ; IF SAME THEN NO DOUBLE STEP
00003E74 808F[CD580100]20    <1>       OR    byte [DSK_STATE+eDI],DBL_STEP ; TURN ON DOUBLE STEP REQUIRED
                             <1> NO_DBL:
00003E7B F8                  <1>       CLC                       ; CLEAR ERROR FLAG
00003E7C C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; READ_ID
                             <1> ;     READ ID FUNCTION.
                             <1> ;
                             <1> ; ON ENTRY:  DI : BIT 2 = HEAD; BITS 1,0 = DRIVE
                             <1> ;
                             <1> ; ON EXIT:   DI : BIT 2 IS RESET, BITS 1,0 = DRIVE
                             <1> ;            @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;-------------------------------------------------------------------------------
                             <1> READ_ID:
00003E7D B8[9A3E0000]        <1>       MOV   eAX, ER_3           ; MOVE NEC OUTPUT ERROR ADDRESS
00003E82 50                  <1>       PUSH  eAX
00003E83 B44A                <1>       MOV   AH,4AH              ; READ ID COMMAND
00003E85 E820010000          <1>       CALL  NEC_OUTPUT          ; TO CONTROLLER
00003E8A 6689F8              <1>       MOV   AX,DI               ; DRIVE # TO AH, HEAD 0
00003E8D 88C4                <1>       MOV   AH,AL
00003E8F E816010000          <1>       CALL  NEC_OUTPUT          ; TO CONTROLLER
00003E94 E80BFEFFFF          <1>       CALL  NEC_TERM            ; WAIT FOR OPERATION, GET STATUS
00003E99 58                  <1>       POP   eAX                 ; THROW AWAY ERROR ADDRESS
                             <1> ER_3:
00003E9A C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; CMOS_TYPE
                             <1> ;     RETURNS DISKETTE TYPE FROM CMOS
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ;
                             <1> ; ON EXIT:   AL = TYPE; CY REFLECTS STATUS
                             <1> ;-------------------------------------------------------------------------------
                             <1>
                             <1> CMOS_TYPE: ; 11/12/2014
00003E9B 8A87[F65C0000]      <1> mov   al, [eDI+fd0_type]
00003EA1 20C0                <1> and   al, al ; 18/12/2014
00003EA3 C3                  <1> retn
                             <1>
                             <1> ;CMOS_TYPE:
                             <1> ;     MOV   AL, CMOS_DIAG       ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS
                             <1> ;     CALL  CMOS_READ           ; GET CMOS STATUS
                             <1> ;     TEST  AL,BAD_BAT+BAD_CKSUM      ; BATTERY GOOD AND CHECKSUM VALID
                             <1> ;     STC                       ; SET CY = 1 INDICATING ERROR FOR RETURN
                             <1> ;     JNZ   short BAD_CM        ; ERROR IF EITHER BIT ON
                             <1> ;     MOV   AL,CMOS_DISKETTE    ; ADDRESS OF DISKETTE BYTE IN CMOS
                             <1> ;     CALL  CMOS_READ           ; GET DISKETTE BYTE
                             <1> ;     OR    DI,DI               ; SEE WHICH DRIVE IN QUESTION
                             <1> ;     JNZ   short TB            ; IF DRIVE 1, DATA IN LOW NIBBLE
                             <1> ;     ROR   AL,4                ; EXCHANGE NIBBLES IF SECOND DRIVE
                             <1> ;TB:
                             <1> ;     AND   AL,0FH              ; KEEP ONLY DRIVE DATA, RESET CY, 0
                             <1> ;BAD_CM:
                             <1> ;     RETn                      ; CY, STATUS OF READ
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; GET_PARM
                             <1> ;     THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE DISK_BASE
                             <1> ;     BLOCK POINTED TO BY THE DATA VARIABLE @DISK_POINTER. A BYTE FROM
                             <1> ;     THAT TABLE IS THEN MOVED INTO AH, THE INDEX OF THAT BYTE BEING
                             <1> ;     THE PARAMETER IN DL.
                             <1> ;
                             <1> ; ON ENTRY:  DL = INDEX OF BYTE TO BE FETCHED
                             <1> ;
                             <1> ; ON EXIT:   AH = THAT BYTE FROM BLOCK
                             <1> ;            AL,DH DESTROYED
                             <1> ;-------------------------------------------------------------------------------
                             <1> GET_PARM:
                             <1>       ;PUSH  DS
00003EA4 56                  <1>       PUSH  eSI
                             <1>       ;SUB  AX,AX               ; DS = 0, BIOS DATA AREA
                             <1>       ;MOV  DS,AX
                             <1>       ;;mov ax, cs
                             <1>       ;;mov ds, ax
                             <1>       ; 08/02/2015 (protected mode modifications, bx -> ebx)
00003EA5 87D3                <1>       XCHG  eDX,eBX                   ; BL = INDEX
                             <1>       ;SUB  BH,BH               ; BX = INDEX
00003EA7 81E3FF000000        <1>       and   ebx, 0FFh
                             <1>       ;LDS  SI, [DISK_POINTER]  ; POINT TO BLOCK
                             <1>       ;
                             <1>       ; 17/12/2014
00003EAD 66A1[E55C0000]      <1>       mov   ax, [cfd] ; current (AL) and previous fd (AH)
00003EB3 38E0                <1>       cmp   al, ah
00003EB5 7425                <1>       je    short gpndc
00003EB7 A2[E65C0000]        <1>       mov   [pfd], al ; current drive -> previous drive
00003EBC 53                  <1>       push  ebx ; 08/02/2015
00003EBD 88C3                <1>       mov   bl, al
                             <1>       ; 11/12/2014
00003EBF 8A83[F65C0000]      <1>       mov   al, [eBX+fd0_type]  ; Drive type (0,1,2,3,4)
                             <1>       ; 18/12/2014
00003EC5 20C0                <1>       and   al, al
00003EC7 7507                <1>       jnz   short gpdtc
00003EC9 BB[CF5C0000]        <1>       mov   ebx, MD_TBL6        ; 1.44 MB param. tbl. (default)
00003ECE EB05                <1>         jmp     short gpdpu
                             <1> gpdtc:
00003ED0 E817F9FFFF          <1>       call  DR_TYPE_CHECK
                             <1>       ; cf = 1 -> eBX points to 1.44MB fd parameter table (default)
                             <1> gpdpu:
00003ED5 891D[6C5C0000]      <1>       mov   [DISK_POINTER], ebx
00003EDB 5B                  <1>       pop   ebx
                             <1> gpndc:
00003EDC 8B35[6C5C0000]      <1>       mov   esi, [DISK_POINTER] ; 08/02/2015, si -> esi
00003EE2 8A241E              <1>       MOV   AH, [eSI+eBX]       ; GET THE WORD
00003EE5 87D3                <1>       XCHG  eDX,eBX                   ; RESTORE BX
00003EE7 5E                  <1>       POP   eSI
                             <1>       ;POP  DS
00003EE8 C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; MOTOR_ON
                             <1> ;     TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE @MOTOR_COUNT
                             <1> ;     IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFH) TO ENSURE
                             <1> ;     THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE
                             <1> ;     MOTOR NEEDED TO BE TURNED ON, THE MULTI-TASKING HOOK FUNCTION
                             <1> ;     (AX=90FDH, INT 15) IS CALLED TELLING THE OPERATING SYSTEM
                             <1> ;     THAT THE BIOS IS ABOUT TO WAIT FOR MOTOR START UP. IF THIS
                             <1> ;     FUNCTION RETURNS WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT
                             <1> ;     HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE
                             <1> ;     THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID
                             <1> ;     NOT WAIT, THE WAIT FUNCTION (AH=086H) IS CALLED TO WAIT THE
                             <1> ;     PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN,
                             <1> ;     IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE
                             <1> ;     WAIT. A TIMER 1 WAIT LOOP WILL THEN DO THE WAIT.
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ; ON EXIT:   AX,CX,DX DESTROYED
                             <1> ;-------------------------------------------------------------------------------
                             <1> MOTOR_ON:
00003EE9 53                  <1>       PUSH  eBX                 ; SAVE REG.
00003EEA E82A000000          <1>       CALL  TURN_ON                   ; TURN ON MOTOR
00003EEF 7226                <1>       JC    short MOT_IS_ON           ; IF CY=1 NO WAIT
```

```
00003EF1 E89BF9FFFF          <1>       CALL   XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
00003EF6 E865F9FFFF          <1>       CALL   XLAT_NEW            ; TRANSLATE STATE TO PRESENT ARCH,
                             <1>       ;CALL  TURN_ON             ; CHECK AGAIN IF MOTOR ON
                             <1>       ;JC    MOT_IS_ON           ; IF NO WAIT MEANS IT IS ON
                             <1> M_WAIT:
00003EFB B20A                <1>       MOV    DL,10               ; GET THE MOTOR WAIT PARAMETER
00003EFD E8A2FFFFFF          <1>       CALL   GET_PARM
                             <1>       ;MOV   AL,AH               ; AL = MOTOR WAIT PARAMETER
                             <1>       ;XOR   AH,AH               ; AX = MOTOR WAIT PARAMETER
                             <1>       ;CMP   AL,8                ; SEE IF AT LEAST A SECOND IS SPECIFIED
00003F02 80FC08              <1>       cmp    ah, 8
                             <1>       ;JAE   short GP2           ; IF YES, CONTINUE
00003F05 7702                <1>       ja     short J13
                             <1>       ;MOV   AL,8                ; ONE SECOND WAIT FOR MOTOR START UP
00003F07 B408                <1>       mov    ah, 8
                             <1>
                             <1> ;-----      AS CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT
                             <1> GP2:
                             <1> ;-----      FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE
                             <1> J13:                             ; WAIT FOR 1/8 SECOND PER (AL)
00003F09 B95E200000          <1>       MOV    eCX,8286            ; COUNT FOR 1/8 SECOND AT 15.085737 US
00003F0E E8DADEFFFF          <1>       CALL   WAITF               ; GO TO FIXED WAIT ROUTINE
                             <1>       ;DEC   AL                  ; DECREMENT TIME VALUE
00003F13 FECC                <1>       dec    ah
00003F15 75F2                <1>       JNZ    short J13           ; ARE WE DONE YET
                             <1> MOT_IS_ON:
00003F17 5B                  <1>       POP    eBX                 ; RESTORE REG.
00003F18 C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; TURN_ON
                             <1> ;     TURN MOTOR ON AND RETURN WAIT STATE.
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ;
                             <1> ; ON EXIT:   CY = 0 MEANS WAIT REQUIRED
                             <1> ;            CY = 1 MEANS NO WAIT REQUIRED
                             <1> ;            AX,BX,CX,DX DESTROYED
                             <1> ;-------------------------------------------------------------------------------
                             <1> TURN_ON:
00003F19 89FB                <1>       MOV    eBX,eDI                   ; BX = DRIVE #
00003F1B 88D9                <1>       MOV    CL,BL               ; CL = DRIVE #
00003F1D C0C304              <1>       ROL    BL,4                ; BL = DRIVE SELECT
00003F20 FA                  <1>       CLI                        ; NO INTERRUPTS WHILE DETERMINING STATUS
00003F21 C605[BF580100]FF    <1>       MOV    byte [MOTOR_COUNT],0FFH   ; ENSURE MOTOR STAYS ON FOR OPERATION
00003F28 A0[BE580100]        <1>       MOV    AL, [MOTOR_STATUS]  ; GET DIGITAL OUTPUT REGISTER REFLECTION
00003F2D 2430                <1>       AND    AL,00110000B        ; KEEP ONLY DRIVE SELECT BITS
00003F2F B401                <1>       MOV    AH,1                ; MASK FOR DETERMINING MOTOR BIT
00003F31 D2E4                <1>       SHL    AH,CL               ; AH = MOTOR ON, A=00000001, B=00000010
                             <1>
                             <1> ;  AL = DRIVE SELECT FROM @MOTOR_STATUS
                             <1> ;  BL = DRIVE SELECT DESIRED
                             <1> ;  AH = MOTOR ON MASK DESIRED
                             <1>
00003F33 38D8                <1>       CMP    AL,BL               ; REQUESTED DRIVE ALREADY SELECTED ?
00003F35 7508                <1>       JNZ    short TURN_IT_ON    ; IF NOT SELECTED JUMP
00003F37 8425[BE580100]      <1>       TEST   AH, [MOTOR_STATUS]  ; TEST MOTOR ON BIT
00003F3D 7535                <1>       JNZ    short NO_MOT_WAIT   ; JUMP IF MOTOR ON AND SELECTED
                             <1>
                             <1> TURN_IT_ON:
00003F3F 08DC                <1>       OR     AH,BL               ; AH = DRIVE SELECT AND MOTOR ON
00003F41 8A3D[BE580100]      <1>       MOV    BH,[MOTOR_STATUS]   ; SAVE COPY OF @MOTOR_STATUS BEFORE
00003F47 80E70F              <1>       AND    BH,00001111B        ; KEEP ONLY MOTOR BITS
00003F4A 8025[BE580100]CF    <1>       AND    byte [MOTOR_STATUS],11001111B ; CLEAR OUT DRIVE SELECT
00003F51 0825[BE580100]      <1>       OR     [MOTOR_STATUS],AH   ; OR IN DRIVE SELECTED AND MOTOR ON
00003F57 A0[BE580100]        <1>       MOV    AL,[MOTOR_STATUS]   ; GET DIGITAL OUTPUT REGISTER REFLECTION
00003F5C 88C3                <1>       MOV    BL,AL               ; BL=@MOTOR_STATUS AFTER, BH=BEFORE
00003F5E 80E30F              <1>       AND    BL,00001111B        ; KEEP ONLY MOTOR BITS
00003F61 FB                  <1>       STI                        ; ENABLE INTERRUPTS AGAIN
00003F62 243F                <1>       AND    AL,00111111B        ; STRIP AWAY UNWANTED BITS
00003F64 C0C004              <1>       ROL    AL,4                ; PUT BITS IN DESIRED POSITIONS
00003F67 0C0C                <1>       OR     AL,00001100B        ; NO RESET, ENABLE DMA/INTERRUPT
00003F69 66BAF203            <1>       MOV    DX,03F2H            ; SELECT DRIVE AND TURN ON MOTOR
00003F6D EE                  <1>       OUT    DX,AL
00003F6E 38FB                <1>       CMP    BL,BH               ; NEW MOTOR TURNED ON ?
                             <1>       ;JZ    short NO_MOT_WAIT   ; NO WAIT REQUIRED IF JUST SELECT
00003F70 7403                <1>       je     short no_mot_w1 ; 27/02/2015
00003F72 F8                  <1>       CLC                        ; (re)SET CARRY MEANING WAIT
00003F73 C3                  <1>       RETn
                             <1>
                             <1> NO_MOT_WAIT:
00003F74 FB                  <1>       sti
                             <1> no_mot_w1: ; 27/02/2015
00003F75 F9                  <1>       STC                        ; SET NO WAIT REQUIRED
                             <1>       ;STI                       ; INTERRUPTS BACK ON
00003F76 C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; HD_WAIT
                             <1> ;     WAIT FOR HEAD SETTLE TIME.
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ;
                             <1> ; ON EXIT:   AX,BX,CX,DX DESTROYED
                             <1> ;-------------------------------------------------------------------------------
                             <1> HD_WAIT:
00003F77 B209                <1>       MOV    DL,9                ; GET HEAD SETTLE PARAMETER
00003F79 E826FFFFFF          <1>       CALL   GET_PARM
00003F7E 08E4                <1>       or     ah, ah ; 17/12/2014
00003F80 7519                <1>       jnz    short DO_WAT
00003F82 F605[BE580100]80    <1>         TEST    byte [MOTOR_STATUS],10000000B ; SEE IF A WRITE OPERATION
                             <1>       ;JZ    short ISNT_WRITE    ; IF NOT, DO NOT ENFORCE ANY VALUES
                             <1>       ;OR    AH,AH               ; CHECK FOR ANY WAIT?
                             <1>       ;JNZ   short DO_WAT        ; IF THERE DO NOT ENFORCE
00003F89 741E                <1>       jz     short HW_DONE
00003F8B B40F                <1>       MOV    AH,HD12_SETTLE            ; LOAD 1.2M HEAD SETTLE MINIMUM
00003F8D 8A87[CD580100]      <1>       MOV    AL,[DSK_STATE+eDI]  ; LOAD STATE
00003F93 24C0                <1>       AND    AL,RATE_MSK         ; KEEP ONLY RATE
00003F95 3C80                <1>       CMP    AL,RATE_250         ; 1.2 M DRIVE ?
00003F97 7502                <1>       JNZ    short DO_WAT        ; DEFAULT HEAD SETTLE LOADED
                             <1> ;GP3:
00003F99 B414                <1>       MOV    AH,HD320_SETTLE           ; USE 320/360 HEAD SETTLE
                             <1> ;     JMP    SHORT DO_WAT
                             <1>
                             <1> ;ISNT_WRITE:
                             <1> ;     OR     AH,AH               ; CHECK FOR NO WAIT
                             <1> ;     JZ     short HW_DONE       ; IF NOT WRITE AND 0 ITS OK
                             <1>
                             <1> ;-----       AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
                             <1> DO_WAT:
                             <1> ;     MOV    AL,AH               ; AL = # MILLISECONDS
                             <1> ;     ;XOR   AH,AH               ; AX = # MILLISECONDS
                             <1> J29:                             ;      1 MILLISECOND LOOP
                             <1>       ;mov   cx, WAIT_FDU_HEAD_SETTLE ; 33 ; 1 ms in 30 micro units.
00003F9B B942000000          <1>       MOV    eCX,66              ; COUNT AT 15.085737 US PER COUNT
00003FA0 E848DEFFFF          <1>       CALL   WAITF               ; DELAY FOR 1 MILLISECOND
                             <1>       ;DEC   AL                  ; DECREMENT THE COUNT
00003FA5 FECC                <1>       dec    ah
00003FA7 75F2                <1>       JNZ    short J29           ; DO AL MILLISECOND # OF TIMES
                             <1> HW_DONE:
00003FA9 C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; NEC_OUTPUT
                             <1> ;     THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING
                             <1> ;     FOR CORRECT DIRECTION AND CONTROLLER READY THIS ROUTINE WILL
                             <1> ;     TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE AMOUNT
                             <1> ;     OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION.
                             <1> ;
                             <1> ; ON ENTRY:  AH = BYTE TO BE OUTPUT
                             <1> ;
                             <1> ; ON EXIT:   CY = 0  SUCCESS
                             <1> ;            CY = 1  FAILURE -- DISKETTE STATUS UPDATED
                             <1> ;                    IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
                             <1> ;                    HIGHER THAN THE CALLER OF NEC OUTPUT. THIS REMOVES THE
                             <1> ;                    REQUIREMENT OF TESTING AFTER EVERY CALL OF NEC_OUTPUT.
                             <1> ;            AX,CX,DX DESTROYED
                             <1> ;-------------------------------------------------------------------------------
                             <1>
                             <1> ; 09/12/2014 [Erdogan Tan]
                             <1> ;     (from 'PS2 Hardware Interface Tech. Ref. May 88', Page 09-05.)
                             <1> ; Diskette Drive Controller Status Register (3F4h)
                             <1> ;     This read only register facilitates the transfer of data between
                             <1> ;     the system microprocessor and the controller.
                             <1> ; Bit 7 - When set to 1, the Data register is ready to transfer data
                             <1> ;       with the system micrprocessor.
                             <1> ; Bit 6 - The direction of data transfer. If this bit is set to 0,
                             <1> ;       the transfer is to the controller.
                             <1> ; Bit 5 - When this bit is set to 1, the controller is in the non-DMA mode.
                             <1> ; Bit 4 - When this bit is set to 1, a Read or Write command is being executed.
                             <1> ; Bit 3 - Reserved.
                             <1> ; Bit 2 - Reserved.
                             <1> ; Bit 1 - When this bit is set to 1, dskette drive 1 is in the seek mode.
                             <1> ; Bit 0 - When this bit is set to 1, dskette drive 1 is in the seek mode.
                             <1>
                             <1> ; Data Register (3F5h)
                             <1> ; This read/write register passes data, commands and parameters, and provides
                             <1> ; diskette status information.
                             <1>
                             <1> NEC_OUTPUT:
                             <1>       ;PUSH  BX                  ; SAVE REG.
00003FAA 66BAF403            <1>       MOV    DX,03F4H            ; STATUS PORT
                             <1>       ;MOV   BL,2                ; HIGH ORDER COUNTER
                             <1>       ;XOR   CX,CX               ; COUNT FOR TIME OUT
                             <1>       ; 16/12/2014
                             <1>       ; waiting for (max.) 0.5 seconds
                             <1>         ;;mov     byte [wait_count], 0 ;; 27/02/2015
                             <1>       ;
                             <1>       ; 17/12/2014
                             <1>       ; Modified from AWARD BIOS 1999 - ADISK.ASM - SEND_COMMAND
                             <1>       ;
                             <1>       ;WAIT_FOR_PORT:     Waits for a bit at a port pointed to by DX to
                             <1>       ;            go on.
                             <1>       ;INPUT:
                             <1>       ;      AH=Mask for isolation bits.
                             <1>       ;      AL=pattern to look for.
                             <1>       ;      DX=Port to test for
                             <1>       ;      BH:CX=Number of memory refresh periods to delay.
                             <1>       ;           (normally 30 microseconds per period.)
                             <1>       ;
                             <1>       ;WFP_SHORT:
                             <1>       ;      Wait for port if refresh cycle is short (15-80 Us range).
                             <1>       ;
                             <1>
                             <1> ;     mov    bl, WAIT_FDU_SEND_HI+1    ; 0+1
                             <1> ;     mov    cx, WAIT_FDU_SEND_LO      ; 16667
00003FAE B91B410000          <1>       mov    ecx, WAIT_FDU_SEND_LH   ; 16667 (27/02/2015)
                             <1> ;
                             <1> ;WFPS_OUTER_LP:
                             <1> ;     ;
                             <1> ;WFPS_CHECK_PORT:
                             <1> J23:
00003FB3 EC                  <1>       IN     AL,DX               ; GET STATUS
00003FB4 24C0                <1>       AND    AL,11000000B        ; KEEP STATUS AND DIRECTION
00003FB6 3C80                <1>       CMP    AL,10000000B        ; STATUS 1 AND DIRECTION 0 ?
00003FB8 7418                <1>       JZ     short J27           ; STATUS AND DIRECTION OK
```

```
                             <1> WFPS_HI:
00003FBA E461                <1>       IN     AL, PORT_B   ;061h ; SYS1 ; wait for hi to lo
00003FBC A810                <1>       TEST   AL,010H                    ; transition on memory
00003FBE 75FA                <1>       JNZ    SHORT WFPS_HI       ; refresh.
                             <1> WFPS_LO:
00003FC0 E461                <1>       IN     AL, PORT_B          ; SYS1
00003FC2 A810                <1>       TEST   AL,010H
00003FC4 74FA                <1>       JZ     SHORT WFPS_LO
                             <1>       ;LOOP  SHORT WFPS_CHECK_PORT
00003FC6 E2EB                <1>       loop   J23    ; 27/02/2015
                             <1> ;     ;
                             <1> ;     dec    bl
                             <1> ;     jnz    short WFPS_OUTER_LP
                             <1> ;     jmp    short WFPS_TIMEOUT  ; fail
                             <1> ;J23:
                             <1> ;     IN     AL,DX               ; GET STATUS
                             <1> ;     AND    AL,11000000B        ; KEEP STATUS AND DIRECTION
                             <1> ;     CMP    AL,10000000B        ; STATUS 1 AND DIRECTION 0 ?
                             <1> ;     JZ     short J27           ; STATUS AND DIRECTION OK
                             <1>       ;LOOP  J23                 ; CONTINUE TILL CX EXHAUSTED
                             <1>       ;DEC   BL                  ; DECREMENT COUNTER
                             <1>       ;JNZ   short J23           ; REPEAT TILL DELAY FINISHED, CX = 0
                             <1>
                             <1>       ;;27/02/2015
                             <1>       ;16/12/2014
                             <1>         ;;cmp     byte [wait_count], 10   ; (10/18.2 seconds)
                             <1>       ;;jb   short J23
                             <1>
                             <1> ;WFPS_TIMEOUT:
                             <1>
                             <1> ;-----       FALL THRU TO ERROR RETURN
                             <1>
00003FC8 800D[C0580100]80    <1>       OR     byte [DSKETTE_STATUS],TIME_OUT
                             <1>       ;POP   BX                  ; RESTORE REG.
00003FCF 58                  <1>       POP    eAX ; 08/02/2015    ; DISCARD THE RETURN ADDRESS
00003FD0 F9                  <1>       STC                        ; INDICATE ERROR TO CALLER
00003FD1 C3                  <1>       RETn
                             <1>
                             <1> ;-----       DIRECTION AND STATUS OK; OUTPUT BYTE
                             <1>
                             <1> J27:
00003FD2 88E0                <1>       MOV    AL,AH               ; GET BYTE TO OUTPUT
00003FD4 6642                <1>       INC    DX                  ; DATA PORT = STATUS PORT + 1
00003FD6 EE                  <1>       OUT    DX,AL               ; OUTPUT THE BYTE
                             <1>       ;;NEWIODELAY  ;; 27/02/2015
                             <1>       ; 27/02/2015
00003FD7 9C                  <1>       PUSHF                      ; SAVE FLAGS
00003FD8 B903000000          <1>       MOV    eCX, 3              ; 30 TO 45 MICROSECONDS WAIT FOR
00003FDD E80BDEFFFF          <1>       CALL   WAITF               ; NEC FLAGS UPDATE CYCLE
00003FE2 9D                  <1>       POPF                       ; RESTORE FLAGS FOR EXIT
                             <1>       ;POP   BX                  ; RESTORE REG
00003FE3 C3                  <1>       RETn                       ; CY = 0 FROM TEST INSTRUCTION
                             <1>
                             <1> ;--------------------------------------------------------------------------------
                             <1> ; SEEK
                             <1> ;     THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE TO THE NAMED
                             <1> ;     TRACK. IF THE DRIVE HAS NOT BEEN ACCESSED SINCE THE DRIVE
                             <1> ;     RESET COMMAND WAS ISSUED, THE DRIVE WILL BE RECALIBRATED.
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ;            CH = TRACK #
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
                             <1> ;            AX,BX,CX DX DESTROYED
                             <1> ;--------------------------------------------------------------------------------
                             <1> SEEK:
00003FE4 89FB                <1>       MOV    eBX,eDI                    ; BX = DRIVE #
00003FE6 B001                <1>       MOV    AL,1                ; ESTABLISH MASK FOR RECALIBRATE TEST
00003FE8 86CB                <1>       XCHG   CL,BL               ; SET DRIVE VALULE INTO CL
00003FEA D2C0                <1>       ROL    AL,CL               ; SHIFT MASK BY THE DRIVE VALUE
00003FEC 86CB                <1>       XCHG   CL,BL               ; RECOVER TRACK VALUE
00003FEE 8405[BD580100]      <1>       TEST   AL,[SEEK_STATUS]    ; TEST FOR RECALIBRATE REQUIRED
00003FF4 7526                <1>       JNZ    short J28A          ; JUMP IF RECALIBRATE NOT REQUIRED
                             <1>
00003FF6 0805[BD580100]      <1>       OR     [SEEK_STATUS],AL    ; TURN ON THE NO RECALIBRATE BIT IN FLAG
00003FFC E862000000          <1>       CALL   RECAL               ; RECALIBRATE DRIVE
00004001 730E                <1>       JNC    short AFT_RECAL            ; RECALIBRATE DONE
                             <1>
                             <1> ;-----       ISSUE RECALIBRATE FOR 80 TRACK DISKETTES
                             <1>
00004003 C605[C0580100]00    <1>       MOV    byte [DSKETTE_STATUS],0    ; CLEAR OUT INVALID STATUS
0000400A E854000000          <1>       CALL   RECAL               ; RECALIBRATE DRIVE
0000400F 7251                <1>       JC     short RB            ; IF RECALIBRATE FAILS TWICE THEN ERROR
                             <1>
                             <1> AFT_RECAL:
00004011 C687[D1580100]00    <1>         MOV     byte [DSK_TRK+eDI],0    ; SAVE NEW CYLINDER AS PRESENT POSITION
00004018 08ED                <1>       OR     CH,CH               ; CHECK FOR SEEK TO TRACK 0
0000401A 743F                <1>       JZ     short DO_WAIT       ; HEAD SETTLE, CY = 0 IF JUMP
                             <1>
                             <1> ;-----       DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
                             <1>
0000401C F687[CD580100]20    <1> J28A: TEST   byte [DSK_STATE+eDI],DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
00004023 7402                <1>       JZ     short _R7           ; SINGLE STEP REQUIRED BYPASS DOUBLE
00004025 D0E5                <1>       SHL    CH,1                ; DOUBLE NUMBER OF STEP TO TAKE
                             <1>
00004027 3AAF[D1580100]      <1> _R7:  CMP    CH, [DSK_TRK+eDI]   ; SEE IF ALREADY AT THE DESIRED TRACK
0000402D 7433                <1>       JE     short RB            ; IF YES, DO NOT NEED TO SEEK
                             <1>
0000402F BA[62400000]        <1>       MOV    eDX, NEC_ERR        ; LOAD RETURN ADDRESS
00004034 52                  <1>       PUSH   eDX ; (*)           ; ON STACK FOR NEC OUTPUT ERROR
00004035 88AF[D1580100]      <1>       MOV    [DSK_TRK+eDI],CH    ; SAVE NEW CYLINDER AS PRESENT POSITION
0000403B B40F                <1>       MOV    AH,0FH              ; SEEK COMMAND TO NEC
0000403D E868FFFFFF          <1>       CALL   NEC_OUTPUT
```

```
00004042 89FB                <1>       MOV    eBX,eDI                  ; BX = DRIVE #
00004044 88DC                <1>       MOV    AH,BL             ; OUTPUT DRIVE NUMBER
00004046 E85FFFFFFF          <1>       CALL   NEC_OUTPUT
0000404B 8AA7[D1580100]      <1>       MOV    AH, [DSK_TRK+eDI]  ; GET CYLINDER NUMBER
00004051 E854FFFFFF          <1>       CALL   NEC_OUTPUT
00004056 E829000000          <1>       CALL   CHK_STAT_2        ; ENDING INTERRUPT AND SENSE STATUS
                             <1>
                             <1> ;-----      WAIT FOR HEAD SETTLE
                             <1>
                             <1> DO_WAIT:
0000405B 9C                  <1>       PUSHF                    ; SAVE STATUS
0000405C E816FFFFFF          <1>       CALL   HD_WAIT                  ; WAIT FOR HEAD SETTLE TIME
00004061 9D                  <1>       POPF                     ; RESTORE STATUS
                             <1> RB:
                             <1> NEC_ERR:
                             <1>       ; 08/02/2015 (code trick here from original IBM PC/AT DISKETTE.ASM)
                             <1>       ; (*) nec_err -> retn (push edx -> pop edx) -> nec_err -> retn
00004062 C3                  <1>       RETn                     ; RETURN TO CALLER
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; RECAL
                             <1> ;     RECALIBRATE DRIVE
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ;
                             <1> ; ON EXIT:   CY REFLECTS STATUS OF OPERATION.
                             <1> ;-------------------------------------------------------------------------------
                             <1> RECAL:
00004063 6651                <1>       PUSH   CX
00004065 B8[81400000]        <1>       MOV    eAX, RC_BACK      ; LOAD NEC_OUTPUT ERROR
0000406A 50                  <1>       PUSH   eAX
0000406B B407                <1>       MOV    AH,07H            ; RECALIBRATE COMMAND
0000406D E838FFFFFF          <1>       CALL   NEC_OUTPUT
00004072 89FB                <1>       MOV    eBX,eDI                  ; BX = DRIVE #
00004074 88DC                <1>       MOV    AH,BL
00004076 E82FFFFFFF          <1>       CALL   NEC_OUTPUT        ; OUTPUT THE DRIVE NUMBER
0000407B E804000000          <1>       CALL   CHK_STAT_2        ; GET THE INTERRUPT AND SENSE INT STATUS
00004080 58                  <1>       POP    eAX               ; THROW AWAY ERROR
                             <1> RC_BACK:
00004081 6659                <1>       POP    CX
00004083 C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; CHK_STAT_2
                             <1> ;     THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER RECALIBRATE,
                             <1> ;     OR SEEK TO THE ADAPTER. THE INTERRUPT IS WAITED FOR, THE
                             <1> ;     INTERRUPT STATUS SENSED, AND THE RESULT RETURNED TO THE CALLER.
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
                             <1> ;-------------------------------------------------------------------------------
                             <1> CHK_STAT_2:
00004084 B8[AC400000]        <1>         MOV     eAX, CS_BACK            ; LOAD NEC_OUTPUT ERROR ADDRESS
00004089 50                  <1>       PUSH   eAX
0000408A E828000000          <1>       CALL   WAIT_INT          ; WAIT FOR THE INTERRUPT
0000408F 721A                <1>       JC     short J34         ; IF ERROR, RETURN IT
00004091 B408                <1>       MOV    AH,08H            ; SENSE INTERRUPT STATUS COMMAND
00004093 E812FFFFFF          <1>       CALL   NEC_OUTPUT
00004098 E84A000000          <1>       CALL   RESULTS                  ; READ IN THE RESULTS
0000409D 720C                <1>       JC     short J34
0000409F A0[C1580100]        <1>       MOV    AL,[NEC_STATUS]          ; GET THE FIRST STATUS BYTE
000040A4 2460                <1>       AND    AL,01100000B      ; ISOLATE THE BITS
000040A6 3C60                <1>       CMP    AL,01100000B      ; TEST FOR CORRECT VALUE
000040A8 7403                <1>       JZ     short J35         ; IF ERROR, GO MARK IT
000040AA F8                  <1>       CLC                      ; GOOD RETURN
                             <1> J34:
000040AB 58                  <1>       POP    eAX               ; THROW AWAY ERROR RETURN
                             <1> CS_BACK:
000040AC C3                  <1>       RETn
                             <1> J35:
000040AD 800D[C0580100]40    <1>       OR     byte [DSKETTE_STATUS], BAD_SEEK
000040B4 F9                  <1>       STC                      ; ERROR RETURN CODE
000040B5 EBF4                <1>       JMP    SHORT J34
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; WAIT_INT
                             <1> ;     THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR A TIME OUT ROUTINE
                             <1> ;     TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE RETURNED
                             <1> ;     IF THE DRIVE IS NOT READY.
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
                             <1> ;-------------------------------------------------------------------------------
                             <1>
                             <1> ; 17/12/2014
                             <1> ; 2.5 seconds waiting !
                             <1> ;(AWARD BIOS - 1999, WAIT_FDU_INT_LOW, WAIT_FDU_INT_HI)
                             <1> ; amount of time to wait for completion interrupt from NEC.
                             <1>
                             <1>
                             <1> WAIT_INT:
000040B7 FB                  <1>       STI                      ; TURN ON INTERRUPTS, JUST IN CASE
000040B8 F8                  <1>       CLC                      ; CLEAR TIMEOUT INDICATOR
                             <1>        ;MOV  BL,10              ; CLEAR THE COUNTERS
                             <1>        ;XOR  CX,CX              ; FOR 2 SECOND WAIT
                             <1>
                             <1>       ; Modification from AWARD BIOS - 1999 (ATORGS.ASM, WAIT
                             <1>       ;
                             <1>       ;WAIT_FOR_MEM:
                             <1>       ;      Waits for a bit at a specified memory location pointed
                             <1>       ;      to by ES:[DI] to become set.
                             <1>       ;INPUT:
                             <1>       ;      AH=Mask to test with.
                             <1>       ;      ES:[DI] = memory location to watch.
                             <1>       ;      BH:CX=Number of memory refresh periods to delay.
```

```
                              <1>       ;          (normally 30 microseconds per period.)
                              <1>
                              <1>       ; waiting for (max.) 2.5 secs in 30 micro units.
                              <1> ;     mov    cx, WAIT_FDU_INT_LO       ; 017798
                              <1> ;;    mov    bl, WAIT_FDU_INT_HI
                              <1> ;     mov    bl, WAIT_FDU_INT_HI + 1
                              <1>       ; 27/02/2015
000040B9 B986450100           <1>       mov    ecx, WAIT_FDU_INT_LH      ; 83334 (2.5 seconds)
                              <1> WFMS_CHECK_MEM:
000040BE F605[BD580100]80     <1>       test   byte [SEEK_STATUS],INT_FLAG ; TEST FOR INTERRUPT OCCURRING
000040C5 7516                 <1>         jnz     short J37
                              <1> WFMS_HI:
000040C7 E461                 <1>       IN     AL,PORT_B  ; 061h   ; SYS1, wait for lo to hi
000040C9 A810                 <1>       TEST   AL,010H                   ; transition on memory
000040CB 75FA                 <1>       JNZ    SHORT WFMS_HI       ; refresh.
                              <1> WFMS_LO:
000040CD E461                 <1>       IN     AL,PORT_B           ;SYS1
000040CF A810                 <1>       TEST   AL,010H
000040D1 74FA                 <1>       JZ     SHORT WFMS_LO
000040D3 E2E9                 <1>         LOOP    WFMS_CHECK_MEM
                              <1> ;WFMS_OUTER_LP:
                              <1> ;;    or     bl, bl              ; check outer counter
                              <1> ;;    jz     short J36A          ; WFMS_TIMEOUT
                              <1> ;     dec    bl
                              <1> ;     jz     short J36A
                              <1> ;     jmp    short WFMS_CHECK_MEM
                              <1>
                              <1>       ;17/12/2014
                              <1>       ;16/12/2014
                              <1> ;        mov     byte [wait_count], 0    ; Reset (INT 08H) counter
                              <1> ;J36:
                              <1> ;     TEST   byte [SEEK_STATUS],INT_FLAG ; TEST FOR INTERRUPT OCCURRING
                              <1> ;     JNZ    short J37
                              <1>       ;16/12/2014
                              <1>       ;LOOP  J36                 ; COUNT DOWN WHILE WAITING
                              <1>       ;DEC   BL                  ; SECOND LEVEL COUNTER
                              <1>       ;JNZ   short J36
                              <1> ;       cmp     byte [wait_count], 46   ; (46/18.2 seconds)
                              <1> ;     jb     short J36
                              <1>
                              <1> ;WFMS_TIMEOUT:
                              <1> ;J36A:
000040D5 800D[C0580100]80     <1>       OR     byte [DSKETTE_STATUS], TIME_OUT ; NOTHING HAPPENED
000040DC F9                   <1>       STC                        ; ERROR RETURN
                              <1> J37:
000040DD 9C                   <1>       PUSHF                      ; SAVE CURRENT CARRY
000040DE 8025[BD580100]7F     <1>       AND    byte [SEEK_STATUS], ~INT_FLAG ; TURN OFF INTERRUPT FLAG
000040E5 9D                   <1>       POPF                       ; RECOVER CARRY
000040E6 C3                   <1>       RETn                       ; GOOD RETURN CODE
                              <1>
                              <1> ;-------------------------------------------------------------------------------
                              <1> ; RESULTS
                              <1> ;     THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER RETURNS
                              <1> ;     FOLLOWING AN INTERRUPT.
                              <1> ;
                              <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
                              <1> ;            AX,BX,CX,DX DESTROYED
                              <1> ;-------------------------------------------------------------------------------
                              <1> RESULTS:
000040E7 57                   <1>       PUSH   eDI
000040E8 BF[C1580100]         <1>       MOV    eDI, NEC_STATUS           ; POINTER TO DATA AREA
000040ED B307                 <1>       MOV    BL,7                ; MAX STATUS BYTES
000040EF 66BAF403             <1>       MOV    DX,03F4H            ; STATUS PORT
                              <1>
                              <1> ;-----      WAIT FOR REQUEST FOR MASTER
                              <1>
                              <1> _R10:
                              <1>       ; 16/12/2014
                              <1>       ; wait for (max) 0.5 seconds
                              <1>       ;MOV   BH,2                ; HIGH ORDER COUNTER
                              <1>       ;XOR   CX,CX               ; COUNTER
                              <1>
                              <1>       ;Time to wait while waiting for each byte of NEC results = .5
                              <1>       ;seconds.  .5 seconds = 500,000 micros.  500,000/30 = 16,667.
                              <1>       ; 27/02/2015
000040F3 B91B410000           <1>       mov    ecx, WAIT_FDU_RESULTS_LH ; 16667
                              <1>       ;mov   cx, WAIT_FDU_RESULTS_LO  ; 16667
                              <1>       ;mov   bh, WAIT_FDU_RESULTS_HI+1 ; 0+1
                              <1>
                              <1> WFPSR_OUTER_LP:
                              <1>       ;
                              <1> WFPSR_CHECK_PORT:
                              <1> J39:                             ; WAIT FOR MASTER
000040F8 EC                   <1>       IN     AL,DX               ; GET STATUS
000040F9 24C0                 <1>       AND    AL,11000000B        ; KEEP ONLY STATUS AND DIRECTION
000040FB 3CC0                 <1>       CMP    AL,11000000B        ; STATUS 1 AND DIRECTION 1 ?
000040FD 7418                 <1>       JZ     short J42           ; STATUS AND DIRECTION OK
                              <1> WFPSR_HI:
000040FF E461                 <1>       IN     AL, PORT_B   ;061h  ; SYS1 ; wait for hi to lo
00004101 A810                 <1>       TEST   AL,010H                   ; transition on memory
00004103 75FA                 <1>       JNZ    SHORT WFPSR_HI            ; refresh.
                              <1> WFPSR_LO:
00004105 E461                 <1>       IN     AL, PORT_B          ; SYS1
00004107 A810                 <1>       TEST   AL,010H
00004109 74FA                 <1>       JZ     SHORT WFPSR_LO
0000410B E2EB                 <1>         LOOP    WFPSR_CHECK_PORT
                              <1>       ;; 27/02/2015
                              <1>       ;;dec  bh
                              <1>       ;;jnz  short WFPSR_OUTER_LP
                              <1>       ;jmp   short WFPSR_TIMEOUT ; fail
                              <1>
                              <1>       ;;mov  byte [wait_count], 0
                              <1> ;J39:                            ; WAIT FOR MASTER
```

```
                             <1> ;     IN     AL,DX               ; GET STATUS
                             <1> ;     AND    AL,11000000B        ; KEEP ONLY STATUS AND DIRECTION
                             <1> ;     CMP    AL,11000000B        ; STATUS 1 AND DIRECTION 1 ?
                             <1> ;     JZ     short J42           ; STATUS AND DIRECTION OK
                             <1>       ;LOOP  J39                 ; LOOP TILL TIMEOUT
                             <1>       ;DEC   BH                  ; DECREMENT HIGH ORDER COUNTER
                             <1>       ;JNZ   short J39           ; REPEAT TILL DELAY DONE
                             <1>       ;
                             <1>       ;;cmp  byte [wait_count], 10  ; (10/18.2 seconds)
                             <1>       ;;jb   short J39
                             <1>
                             <1> ;WFPSR_TIMEOUT:
0000410D 800D[C0580100]80    <1>       OR     byte [DSKETTE_STATUS],TIME_OUT
00004114 F9                  <1>       STC                        ; SET ERROR RETURN
00004115 EB29                <1>       JMP    SHORT POPRES        ; POP REGISTERS AND RETURN
                             <1>
                             <1> ;-----       READ IN THE STATUS
                             <1>
                             <1> J42:
00004117 EB00                <1>       JMP    $+2                 ; I/O DELAY
00004119 6642                <1>       INC    DX                  ; POINT AT DATA PORT
0000411B EC                  <1>       IN     AL,DX               ; GET THE DATA
                             <1>       ; 16/12/2014
                             <1>       NEWIODELAY
0000411C E6EB                <2>  out 0ebh,al
0000411E 8807                <1>         MOV     [eDI],AL                 ; STORE THE BYTE
00004120 47                  <1>       INC    eDI                 ; INCREMENT THE POINTER
                             <1>       ; 16/12/2014
                             <1> ;     push   cx
                             <1> ;     mov    cx, 30
                             <1> ;wdw2:
                             <1> ;     NEWIODELAY
                             <1> ;     loop   wdw2
                             <1> ;     pop    cx
                             <1>
00004121 B903000000          <1>       MOV    eCX,3               ; MINIMUM 24 MICROSECONDS FOR NEC
00004126 E8C2DCFFFF          <1>       CALL   WAITF               ; WAIT 30 TO 45 MICROSECONDS
0000412B 664A                <1>       DEC    DX                  ; POINT AT STATUS PORT
0000412D EC                  <1>       IN     AL,DX               ; GET STATUS
                             <1>       ; 16/12/2014
                             <1>       NEWIODELAY
0000412E E6EB                <2>  out 0ebh,al
                             <1>       ;
00004130 A810                <1>       TEST   AL,00010000B        ; TEST FOR NEC STILL BUSY
00004132 740C                <1>       JZ     short POPRES        ; RESULTS DONE ?
                             <1>
00004134 FECB                <1>       DEC    BL                  ; DECREMENT THE STATUS COUNTER
00004136 75BB                <1>         JNZ     short _R10               ; GO BACK FOR MORE
00004138 800D[C0580100]20    <1>       OR     byte [DSKETTE_STATUS],BAD_NEC ; TOO MANY STATUS BYTES
0000413F F9                  <1>       STC                        ; SET ERROR FLAG
                             <1>
                             <1> ;-----       RESULT OPERATION IS DONE
                             <1> POPRES:
00004140 5F                  <1>       POP    eDI
00004141 C3                  <1>       RETn                       ; RETURN WITH CARRY SET
                             <1>
                             <1> ;--------------------------------------------------------------------------------
                             <1> ; READ_DSKCHNG
                             <1> ;     READS THE STATE OF THE DISK CHANGE LINE.
                             <1> ;
                             <1> ; ON ENTRY: DI = DRIVE #
                             <1> ;
                             <1> ; ON EXIT:  DI = DRIVE #
                             <1> ;            ZF = 0 : DISK CHANGE LINE INACTIVE
                             <1> ;            ZF = 1 : DISK CHANGE LINE ACTIVE
                             <1> ;            AX,CX,DX DESTROYED
                             <1> ;--------------------------------------------------------------------------------
                             <1> READ_DSKCHNG:
00004142 E8A2FDFFFF          <1>       CALL   MOTOR_ON            ; TURN ON THE MOTOR IF OFF
00004147 66BAF703            <1>       MOV    DX,03F7H            ; ADDRESS DIGITAL INPUT REGISTER
0000414B EC                  <1>       IN     AL,DX               ; INPUT DIGITAL INPUT REGISTER
0000414C A880                <1>       TEST   AL,DSK_CHG          ; CHECK FOR DISK CHANGE LINE ACTIVE
0000414E C3                  <1>       RETn                       ; RETURN TO CALLER WITH ZERO FLAG SET
                             <1>
                             <1> ;--------------------------------------------------------------------------------
                             <1> ; DRIVE_DET
                             <1> ;     DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND
                             <1> ;     UPDATES STATE INFORMATION ACCORDINGLY.
                             <1> ; ON ENTRY: DI = DRIVE #
                             <1> ;--------------------------------------------------------------------------------
                             <1> DRIVE_DET:
0000414F E895FDFFFF          <1>       CALL   MOTOR_ON            ; TURN ON MOTOR IF NOT ALREADY ON
00004154 E80AFFFFFF          <1>       CALL   RECAL               ; RECALIBRATE DRIVE
00004159 7251                <1>       JC     short DD_BAC        ; ASSUME NO DRIVE PRESENT
0000415B B530                <1>       MOV    CH,TRK_SLAP         ; SEEK TO TRACK 48
0000415D E882FEFFFF          <1>       CALL   SEEK
00004162 7248                <1>       JC     short DD_BAC        ; ERROR NO DRIVE
00004164 B50B                <1>       MOV    CH,QUIET_SEEK+1           ; SEEK TO TRACK 10
                             <1> SK_GIN:
00004166 FECD                <1>       DEC    CH                  ; DECREMENT TO NEXT TRACK
00004168 6651                <1>       PUSH   CX                  ; SAVE TRACK
0000416A E875FEFFFF          <1>       CALL   SEEK
0000416F 723C                <1>       JC     short POP_BAC       ; POP AND RETURN
00004171 B8[AD410000]        <1>       MOV    eAX, POP_BAC        ; LOAD NEC OUTPUT ERROR ADDRESS
00004176 50                  <1>       PUSH   eAX
00004177 B404                <1>       MOV    AH,SENSE_DRV_ST           ; SENSE DRIVE STATUS COMMAND BYTE
00004179 E82CFEFFFF          <1>       CALL   NEC_OUTPUT          ; OUTPUT TO NEC
0000417E 6689F8              <1>       MOV    AX,DI               ; AL = DRIVE
00004181 88C4                <1>       MOV    AH,AL               ; AH = DRIVE
00004183 E822FEFFFF          <1>       CALL   NEC_OUTPUT          ; OUTPUT TO NEC
00004188 E85AFFFFFF          <1>       CALL   RESULTS                   ; GO GET STATUS
0000418D 58                  <1>       POP    eAX                 ; THROW AWAY ERROR ADDRESS
0000418E 6659                <1>       POP    CX                  ; RESTORE TRACK
```

```
00004190 F605[C1580100]10    <1>       TEST   byte [NEC_STATUS], HOME   ; TRACK 0 ?
00004197 74CD                <1>       JZ     short SK_GIN       ; GO TILL TRACK 0
00004199 08ED                <1>       OR     CH,CH              ; IS HOME AT TRACK 0
0000419B 7408                <1>       JZ     short IS_80        ; MUST BE 80 TRACK DRIVE
                             <1> 
                             <1> ;     DRIVE IS A 360; SET DRIVE TO DETERMINED;
                             <1> ;     SET MEDIA TO DETERMINED AT RATE 250.
                             <1> 
0000419D 808F[CD580100]94    <1>       OR     byte [DSK_STATE+eDI], DRV_DET+MED_DET+RATE_250
000041A4 C3                  <1>       RETn                      ; ALL INFORMATION SET
                             <1> IS_80:
000041A5 808F[CD580100]01    <1>       OR     byte [DSK_STATE+eDI], TRK_CAPA ; SETUP 80 TRACK CAPABILITY
                             <1> DD_BAC:
000041AC C3                  <1>       RETn
                             <1> POP_BAC:
000041AD 6659                <1>       POP    CX                 ; THROW AWAY
000041AF C3                  <1>       RETn
                             <1> 
                             <1> fdc_int:
                             <1>         ; 30/07/2015
                             <1>         ; 16/02/2015
                             <1> ;int_0Eh: ; 11/12/2014
                             <1> 
                             <1> ;--- HARDWARE INT 0EH -- ( IRQ LEVEL  6 ) ---------------------------------------
                             <1> ; DISK_INT
                             <1> ;     THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.
                             <1> ;
                             <1> ; ON EXIT:   THE INTERRUPT FLAG IS SET IN @SEEK_STATUS.
                             <1> ;--------------------------------------------------------------------------------
                             <1> DISK_INT_1:
                             <1> 
000041B0 6650                <1>       PUSH   AX                 ; SAVE WORK REGISTER
000041B2 1E                  <1>       push   ds
000041B3 66B81000            <1>       mov    ax, KDATA
000041B7 8ED8                <1>       mov    ds, ax
000041B9 800D[BD580100]80    <1>         OR      byte [SEEK_STATUS], INT_FLAG ; TURN ON INTERRUPT OCCURRED
000041C0 B020                <1>       MOV     AL,EOI                  ; END OF INTERRUPT MARKER
000041C2 E620                <1>       OUT    INTA00,AL          ; INTERRUPT CONTROL PORT
000041C4 1F                  <1>       pop    ds
000041C5 6658                <1>       POP    AX                 ; RECOVER REGISTER
000041C7 CF                  <1>       IRETd                     ; RETURN FROM INTERRUPT
                             <1> 
                             <1> ;--------------------------------------------------------------------------------
                             <1> ; DSKETTE_SETUP
                             <1> ;     THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE OF
                             <1> ;     DISKETTE DRIVES ARE ATTACH TO THE SYSTEM.
                             <1> ;--------------------------------------------------------------------------------
                             <1> 
                             <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> 
                             <1> DSKETTE_SETUP:
                             <1>       ;PUSH  AX                 ; SAVE REGISTERS
                             <1>       ;PUSH  BX
                             <1>       ;PUSH  CX
000041C8 52                  <1>       PUSH   eDX
                             <1>       ;PUSH  DI
                             <1>       ;;PUSH DS
                             <1>       ; 14/12/2014
                             <1>       ;mov   word [DISK_POINTER], MD_TBL6
                             <1>       ;mov   [DISK_POINTER+2], cs
                             <1>       ;
                             <1>       ;OR    byte [RTC_WAIT_FLAG], 1   ; NO RTC WAIT, FORCE USE OF LOOP
000041C9 31FF                <1>       XOR    eDI,eDI                   ; INITIALIZE DRIVE POINTER
000041CB 66C705[CD580100]00- <1>       MOV    WORD [DSK_STATE],0  ; INITIALIZE STATES
000041D3 00                  <1> 
000041D4 8025[C8580100]33    <1>       AND    byte [LASTRATE],~(STRT_MSK+SEND_MSK) ; CLEAR START & SEND
000041DB 800D[C8580100]C0    <1>       OR     byte [LASTRATE],SEND_MSK ; INITIALIZE SENT TO IMPOSSIBLE
000041E2 C605[BD580100]00    <1>       MOV    byte [SEEK_STATUS],0      ; INDICATE RECALIBRATE NEEDED
000041E9 C605[BF580100]00    <1>       MOV    byte [MOTOR_COUNT],0      ; INITIALIZE MOTOR COUNT
000041F0 C605[BE580100]00    <1>       MOV    byte [MOTOR_STATUS],0     ; INITIALIZE DRIVES TO OFF STATE
000041F7 C605[C0580100]00    <1>       MOV    byte [DSKETTE_STATUS],0   ; NO ERRORS
                             <1>       ;
                             <1>       ; 28/02/2015
                             <1>       ;mov   word [cfd], 100h
000041FE E848F2FFFF          <1>       call   DSK_RESET
00004203 5A                  <1>       pop    edx
00004204 F8                  <1>       clc    ; 29/05/2016
00004205 C3                  <1>       retn
                             <1> 
                             <1> ;SUP0:
                             <1> ;     CALL   DRIVE_DET          ; DETERMINE DRIVE
                             <1> ;     CALL   XLAT_OLD           ; TRANSLATE STATE TO COMPATIBLE MODE
                             <1> ;     ; 02/01/2015
                             <1> ;     ;INC   DI                 ; POINT TO NEXT DRIVE
                             <1> ;     ;CMP   DI,MAX_DRV         ; SEE IF DONE
                             <1> ;     ;JNZ   short SUP0         ; REPEAT FOR EACH ORIVE
                             <1> ;       cmp     byte [fd1_type], 0
                             <1> ;     jna    short sup1
                             <1> ;     or     di, di
                             <1> ;     jnz    short sup1
                             <1> ;     inc    di
                             <1> ;       jmp     short SUP0
                             <1> ;sup1:
                             <1> ;     MOV    byte [SEEK_STATUS],0      ; FORCE RECALIBRATE
                             <1> ;     ;AND   byte [RTC_WAIT_FLAG],0FEH ; ALLOW FOR RTC WAIT
                             <1> ;     CALL   SETUP_END          ; VARIOUS CLEANUPS
                             <1> ;     ;;POP  DS                 ; RESTORE CALLERS REGISTERS
                             <1> ;     ;POP   DI
                             <1> ;     POP    eDX
                             <1> ;     ;POP   CX
                             <1> ;     ;POP   BX
                             <1> ;     ;POP   AX
                             <1> ;     RETn
```

```
                                  <1>
                                  <1> ;///////////////////////////////////////////////////////
                                  <1> ;; END OF DISKETTE I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  <1> ;
                                  <1>
                                  <1> int13h: ; 21/02/2015
00004206 9C                       <1>       pushfd
00004207 0E                       <1>       push  cs
00004208 E843010000               <1>       call  DISK_IO
0000420D C3                       <1>       retn
                                  <1>
                                  <1> ;;;;;; DISK I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; 21/02/2015 ;;;
                                  <1> ;///////////////////////////////////////////////////////////////////////
                                  <1>
                                  <1> ; DISK I/O - Erdogan Tan (Retro UNIX 386 v1 project)
                                  <1> ; 18/02/2016
                                  <1> ; 17/02/2016
                                  <1> ; 23/02/2015
                                  <1> ; 21/02/2015 (unix386.s)
                                  <1> ; 22/12/2014 - 14/02/2015 (dsectrm2.s)
                                  <1> ;
                                  <1> ; Original Source Code:
                                  <1> ; DISK ----- 09/25/85 FIXED DISK BIOS
                                  <1> ; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)
                                  <1> ;
                                  <1> ; Modifications: by reference of AWARD BIOS 1999 (D1A0622)
                                  <1> ;              Source Code - ATORGS.ASM, AHDSK.ASM
                                  <1> ;
                                  <1>
                                  <1>
                                  <1> ;The wait for controller to be not busy is 10 seconds.
                                  <1> ;10,000,000 / 30 = 333,333.  333,333 decimal = 051615h
                                  <1> ;;WAIT_HDU_CTLR_BUSY_LO   equ    1615h
                                  <1> ;;WAIT_HDU_CTLR_BUSY_HI   equ      05h
                                  <1> WAIT_HDU_CTRL_BUSY_LH     equ    51615h  ;21/02/2015
                                  <1>
                                  <1> ;The wait for controller to issue completion interrupt is 10 seconds.
                                  <1> ;10,000,000 / 30 = 333,333.  333,333 decimal = 051615h
                                  <1> ;;WAIT_HDU_INT_LO  equ    1615h
                                  <1> ;;WAIT_HDU_INT_HI  equ      05h
                                  <1> WAIT_HDU_INT_LH           equ    51615h ; 21/02/2015
                                  <1>
                                  <1> ;The wait for Data request on read and write longs is
                                  <1> ;2000 us. (?)
                                  <1> ;;WAIT_HDU_DRQ_LO  equ    1000   ; 03E8h
                                  <1> ;;WAIT_HDU_DRQ_HI  equ    0
                                  <1> WAIT_HDU_DRQ_LH           equ    1000   ; 21/02/2015
                                  <1>
                                  <1> ; Port 61h (PORT_B)
                                  <1> SYS1         equ    61h    ; PORT_B  (diskette.inc)
                                  <1>
                                  <1> ; 23/12/2014
                                  <1> %define CMD_BLOCK       eBP-8  ; 21/02/2015
                                  <1>
                                  <1>
                                  <1> ;--- INT 13H -------------------------------------------------------------------
                                  <1> ;                                                                 :
                                  <1> ; FIXED DISK I/O INTERFACE                                               :
                                  <1> ;                                                                 :
                                  <1> ;     THIS INTERFACE PROVIDES ACCESS TO 5 1/4" FIXED DISKS THROUGH           :
                                  <1> ;     THE IBM FIXED DISK CONTROLLER.                                   :
                                  <1> ;                                                                 :
                                  <1> ;     THE  BIOS  ROUTINES  ARE  MEANT  TO  BE  ACCESSED  THROUGH        :
                                  <1> ;     SOFTWARE  INTERRUPTS  ONLY.    ANY  ADDRESSES  PRESENT     IN          :
                                  <1> ;     THESE  LISTINGS  ARE  INCLUDED    ONLY  FOR  COMPLETENESS,         :
                                  <1> ;     NOT  FOR  REFERENCE.  APPLICATIONS   WHICH  REFERENCE  ANY         :
                                  <1> ;     ABSOLUTE  ADDRESSES  WITHIN  THE  CODE SEGMENTS  OF  BIOS          :
                                  <1> ;     VIOLATE  THE  STRUCTURE  AND  DESIGN  OF  BIOS.                  :
                                  <1> ;                                                                 :
                                  <1> ;----------------------------------------------------------------------------:
                                  <1> ;                                                                 :
                                  <1> ; INPUT  (AH)= HEX COMMAND VALUE                                     :
                                  <1> ;                                                                 :
                                  <1> ;     (AH)= 00H  RESET DISK (DL = 80H,81H) / DISKETTE                  :
                                  <1> ;     (AH)= 01H  READ THE STATUS OF THE LAST DISK OPERATION INTO (AL)       :
                                  <1> ;                NOTE: DL < 80H - DISKETTE                             :
                                  <1> ;                      DL > 80H - DISK                           :
                                  <1> ;     (AH)= 02H  READ THE DESIRED SECTORS INTO MEMORY                  :
                                  <1> ;     (AH)= 03H  WRITE THE DESIRED SECTORS FROM MEMORY                 :
                                  <1> ;     (AH)= 04H  VERIFY THE DESIRED SECTORS                            :
                                  <1> ;     (AH)= 05H  FORMAT THE DESIRED TRACK                              :
                                  <1> ;     (AH)= 06H  UNUSED                                          :
                                  <1> ;     (AH)= 07H  UNUSED                                          :
                                  <1> ;     (AH)= 08H  RETURN THE CURRENT DRIVE PARAMETERS                           :
                                  <1> ;     (AH)= 09H  INITIALIZE DRIVE PAIR CHARACTERISTICS                 :
                                  <1> ;                INTERRUPT 41 POINTS TO DATA BLOCK FOR DRIVE 0         :
                                  <1> ;                INTERRUPT 46 POINTS TO DATA BLOCK FOR DRIVE 1         :
                                  <1> ;     (AH)= 0AH  READ LONG                                             :
                                  <1> ;     (AH)= 0BH  WRITE LONG  (READ & WRITE LONG ENCOMPASS 512 + 4 BYTES ECC) :
                                  <1> ;     (AH)= 0CH  SEEK                                            :
                                  <1> ;     (AH)= 0DH  ALTERNATE DISK RESET (SEE DL)                         :
                                  <1> ;     (AH)= 0EH  UNUSED                                          :
                                  <1> ;     (AH)= 0FH  UNUSED                                          :
                                  <1> ;     (AH)= 10H  TEST DRIVE READY                                      :
                                  <1> ;     (AH)= 11H  RECALIBRATE                                           :
                                  <1> ;     (AH)= 12H  UNUSED                                          :
                                  <1> ;     (AH)= 13H  UNUSED                                          :
                                  <1> ;     (AH)= 14H  CONTROLLER INTERNAL DIAGNOSTIC                        :
                                  <1> ;     (AH)= 15H  READ DASD TYPE                                  :
                                  <1> ;                                                                 :
                                  <1> ;-----------------------------------------------------------------------------
                                  <1> ;                                                                 :
                                  <1> ;     REGISTERS USED FOR FIXED DISK OPERATIONS                         :
```

```
<1> ;                                                                 :
<1> ;           (DL)   -  DRIVE NUMBER     (80H-81H FOR DISK. VALUE CHECKED)  :
<1> ;           (DH)   -  HEAD NUMBER          (0-15 ALLOWED, NOT VALUE CHECKED)  :
<1> ;           (CH)   -  CYLINDER NUMBER  (0-1023, NOT VALUE CHECKED)(SEE CL):
<1> ;           (CL)   -  SECTOR NUMBER    (1-17, NOT VALUE CHECKED)        :
<1> ;                                                                 :
<1> ;                     NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED     :
<1> ;                          IN THE HIGH 2 BITS OF THE CL REGISTER        :
<1> ;                          (10 BITS TOTAL)                        :
<1> ;                                                                 :
<1> ;           (AL)   -  NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H,    :
<1> ;                                     FOR READ/WRITE LONG 1-79H)       :
<1> ;                                                                 :
<1> ;           (ES:BX) -  ADDRESS OF BUFFER FOR READS AND WRITES,          :
<1> ;                     (NOT REQUIRED FOR VERIFY)                     :
<1> ;                                                                 :
<1> ;           FORMAT (AH=5) ES:BX POINTS TO A 512 BYTE BUFFER. THE FIRST     :
<1> ;                     2*(SECTORS/TRACK) BYTES CONTAIN F,N FOR EACH SECTOR.:
<1> ;                     F = 00H FOR A GOOD SECTOR                     :
<1> ;                         80H FOR A BAD SECTOR                      :
<1> ;                     N = SECTOR NUMBER                             :
<1> ;                     FOR AN INTERLEAVE OF 2 AND 17 SECTORS/TRACK             :
<1> ;                     THE TABLE SHOULD BE:                        :
<1> ;                                                                 :
<1> ;              DB     00H,01H,00H,0AH,00H,02H,00H,0BH,00H,03H,00H,0CH     :
<1> ;              DB     00H,04H,00H,0DH,00H,05H,00H,0EH,00H,06H,00H,0FH     :
<1> ;              DB     00H,07H,00H,10H,00H,08H,00H,11H,00H,09H           :
<1> ;                                                                 :
<1> ;-------------------------------------------------------------------------------
<1>
<1> ;-------------------------------------------------------------------------------
<1> ; OUTPUT                                                          :
<1> ;     AH = STATUS OF CURRENT OPERATION                            :
<1> ;          STATUS BITS ARE DEFINED IN THE EQUATES BELOW             :
<1> ;     CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN)                  :
<1> ;     CY = 1 FAILED OPERATION (AH HAS ERROR REASON)                 :
<1> ;                                                                 :
<1> ;     NOTE:  ERROR 11H  INDICATES THAT THE DATA READ HAD A RECOVERABLE      :
<1> ;            ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM.  THE DATA      :
<1> ;            IS PROBABLY GOOD,   HOWEVER THE BIOS ROUTINE INDICATES AN      :
<1> ;            ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE      :
<1> ;            FOR ITSELF.  THE  ERROR  MAY  NOT  RECUR  IF  THE DATA IS      :
<1> ;            REWRITTEN.                                           :
<1> ;                                                                 :
<1> ;     IF DRIVE PARAMETERS WERE REQUESTED (DL >= 80H),               :
<1> ;        INPUT:                                                   :
<1> ;          (DL) = DRIVE NUMBER                                    :
<1> ;          ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)                                                 :

<1> ;          EBX = Buffer address for fixed disk parameters table (32 bytes)   :
<1> ;        OUTPUT:                                                  :
<1> ;          (DL) = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (1-2)  :
<1> ;               (CONTROLLER CARD ZERO TALLY ONLY)                   :
<1> ;          (DH) = MAXIMUM USEABLE VALUE FOR HEAD NUMBER             :
<1> ;          (CH) = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER                :
<1> ;          (CL) = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER           :
<1> ;               AND CYLINDER NUMBER HIGH BITS                       :
<1> ;                                                                 :
<1> ;     IF READ DASD TYPE WAS REQUESTED,                            :
<1> ;                                                                 :
<1> ;     AH = 0 - NOT PRESENT                                          :
<1> ;          1 - DISKETTE - NO CHANGE LINE AVAILABLE                  :
<1> ;          2 - DISKETTE - CHANGE LINE AVAILABLE                     :
<1> ;          3 - FIXED DISK                                         :
<1> ;                                                                 :
<1> ;     CX,DX = NUMBER OF 512 BYTE BLOCKS WHEN AH = 3                 :
<1> ;                                                                 :
<1> ;     REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN        :
<1> ;     INFORMATION.                                                :
<1> ;                                                                 :
<1> ;     NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE        :
<1> ;            ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION.          :
<1> ;                                                                 :
<1> ;-------------------------------------------------------------------------------
<1>
<1> SENSE_FAIL   EQU    0FFH          ; NOT IMPLEMENTED
<1> NO_ERR              EQU    0E0H          ; STATUS ERROR/ERROR REGISTER=0
<1> WRITE_FAULT  EQU    0CCH          ; WRITE FAULT ON SELECTED DRIVE
<1> UNDEF_ERR    EQU    0BBH          ; UNDEFINED ERROR OCCURRED
<1> NOT_RDY      EQU    0AAH          ; DRIVE NOT READY
<1> TIME_OUT     EQU    80H           ; ATTACHMENT FAILED TO RESPOND
<1> BAD_SEEK     EQU    40H           ; SEEK OPERATION FAILED
<1> BAD_CNTLR    EQU    20H           ; CONTROLLER HAS FAILED
<1> DATA_CORRECTED      EQU    11H           ; ECC CORRECTED DATA ERROR
<1> BAD_ECC      EQU    10H           ; BAD ECC ON DISK READ
<1> BAD_TRACK    EQU    0BH           ; NOT IMPLEMENTED
<1> BAD_SECTOR   EQU    0AH           ; BAD SECTOR FLAG DETECTED
<1> ;DMA_BOUNDARY       EQU    09H           ; DATA EXTENDS TOO FAR
<1> INIT_FAIL    EQU    07H           ; DRIVE PARAMETER ACTIVITY FAILED
<1> BAD_RESET    EQU    05H           ; RESET FAILED
<1> ;RECORD_NOT_FND     EQU    04H           ; REQUESTED SECTOR NOT FOUND
<1> ;BAD_ADDR_MARK      EQU    02H           ; ADDRESS MARK NOT FOUND
<1> ;BAD_CMD     EQU    01H           ; BAD COMMAND PASSED TO DISK I/O
<1>
<1> ;---------------------------------------------------------
<1> ;                                             :
<1> ; FIXED DISK PARAMETER TABLE                         :
<1> ;  -  THE TABLE IS COMPOSED OF A BLOCK DEFINED AS:  :
<1> ;                                             :
<1> ;  +0 (1 WORD) - MAXIMUM NUMBER OF CYLINDERS         :
<1> ;  +2 (1 BYTE) - MAXIMUM NUMBER OF HEADS             :
<1> ;  +3 (1 WORD) - NOT USED/SEE PC-XT                  :
<1> ;  +5 (1 WORD) - STARTING WRITE PRECOMPENSATION CYL :
```

```
<1> ;  +7 (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH      :
<1> ;  +8 (1 BYTE) - CONTROL BYTE                       :
<1> ;               BIT   7 DISABLE RETRIES -OR-   :
<1> ;               BIT   6 DISABLE RETRIES        :
<1> ;               BIT   3 MORE THAN 8 HEADS             :
<1> ;  +9 (3 BYTES)- NOT USED/SEE PC-XT                 :
<1> ; +12 (1 WORD) - LANDING ZONE                  :
<1> ; +14 (1 BYTE) - NUMBER OF SECTORS/TRACK            :
<1> ; +15 (1 BYTE) - RESERVED FOR FUTURE USE            :
<1> ;                                              :
<1> ;      - TO DYNAMICALLY DEFINE A SET OF PARAMETERS  :
<1> ;        BUILD A TABLE FOR UP TO 15 TYPES AND PLACE :
<1> ;        THE CORRESPONDING VECTOR INTO INTERRUPT 41 :
<1> ;        FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1.  :
<1> ;                                              :
<1> ;-----------------------------------------------------
<1>
<1> ;-----------------------------------------------------
<1> ;                                              :
<1> ; HARDWARE SPECIFIC VALUES                            :
<1> ;                                              :
<1> ;  -  CONTROLLER I/O PORT                      :
<1> ;                                              :
<1> ;     > WHEN READ FROM:                        :
<1> ;     HF_PORT+0 - READ DATA (FROM CONTROLLER TO CPU)       :
<1> ;     HF_PORT+1 - GET ERROR REGISTER                    :
<1> ;     HF_PORT+2 - GET SECTOR COUNT                      :
<1> ;     HF_PORT+3 - GET SECTOR NUMBER                     :
<1> ;     HF_PORT+4 - GET CYLINDER LOW                      :
<1> ;     HF_PORT+5 - GET CYLINDER HIGH (2 BITS)            :
<1> ;     HF_PORT+6 - GET SIZE/DRIVE/HEAD          :
<1> ;     HF_PORT+7 - GET STATUS REGISTER          :
<1> ;                                              :
<1> ;     > WHEN WRITTEN TO:                       :
<1> ;     HF_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER) :
<1> ;     HF_PORT+1 - SET PRECOMPENSATION CYLINDER          :
<1> ;     HF_PORT+2 - SET SECTOR COUNT                      :
<1> ;     HF_PORT+3 - SET SECTOR NUMBER                     :
<1> ;     HF_PORT+4 - SET CYLINDER LOW                      :
<1> ;     HF_PORT+5 - SET CYLINDER HIGH (2 BITS)            :
<1> ;     HF_PORT+6 - SET SIZE/DRIVE/HEAD          :
<1> ;     HF_PORT+7 - SET COMMAND REGISTER         :
<1> ;                                              :
<1> ;-----------------------------------------------------
<1>
<1> ;HF_PORT     EQU    01F0H  ; DISK PORT
<1> ;HF1_PORT    equ    0170h
<1> ;HF_REG_PORT EQU    03F6H
<1> ;HF1_REG_PORT      equ    0376h
<1>
<1> HDC1_BASEPORT      equ    1F0h
<1> HDC2_BASEPORT      equ    170h
<1>
<1> align 2
<1>
<1> ;-----             STATUS REGISTER
<1>
<1> ST_ERROR     EQU    00000001B    ;
<1> ST_INDEX     EQU    00000010B    ;
<1> ST_CORRCTD   EQU    00000100B    ; ECC CORRECTION SUCCESSFUL
<1> ST_DRQ             EQU    00001000B    ;
<1> ST_SEEK_COMPL      EQU    00010000B    ; SEEK COMPLETE
<1> ST_WRT_FLT   EQU    00100000B    ; WRITE FAULT
<1> ST_READY     EQU    01000000B    ;
<1> ST_BUSY      EQU    10000000B    ;
<1>
<1> ;-----             ERROR REGISTER
<1>
<1> ERR_DAM      EQU    00000001B    ; DATA ADDRESS MARK NOT FOUND
<1> ERR_TRK_0    EQU    00000010B    ; TRACK 0 NOT FOUND ON RECAL
<1> ERR_ABORT    EQU    00000100B    ; ABORTED COMMAND
<1> ;            EQU    00001000B    ; NOT USED
<1> ERR_ID             EQU    00010000B    ; ID NOT FOUND
<1> ;            EQU    00100000B    ; NOT USED
<1> ERR_DATA_ECC EQU    01000000B
<1> ERR_BAD_BLOCK      EQU    10000000B
<1>
<1>
<1> RECAL_CMD    EQU    00010000B    ; DRIVE RECAL(10H)
<1> READ_CMD     EQU    00100000B    ;      READ   (20H)
<1> WRITE_CMD    EQU    00110000B    ;      WRITE  (30H)
<1> VERIFY_CMD   EQU    01000000B    ;      VERIFY (40H)
<1> FMTTRK_CMD   EQU    01010000B    ; FORMAT TRACK      (50H)
<1> INIT_CMD     EQU    01100000B    ;   INITIALIZE      (60H)
<1> SEEK_CMD     EQU    01110000B    ;      SEEK   (70H)
<1> DIAG_CMD     EQU    10010000B    ; DIAGNOSTIC  (90H)
<1> SET_PARM_CMD EQU    10010001B    ; DRIVE PARMS(91H)
<1> NO_RETRIES   EQU    00000001B    ; CHD MODIFIER      (01H)
<1> ECC_MODE     EQU    00000010B    ; CMD MODIFIER      (02H)
<1> BUFFER_MODE  EQU    00001000B    ; CMD MODIFIER      (08H)
<1>
<1> ;MAX_FILE    EQU    2
<1> ;S_MAX_FILE  EQU    2
<1> MAX_FILE     equ    4            ; 22/12/2014
<1> S_MAX_FILE   equ    4            ; 22/12/2014
<1>
<1> DELAY_1      EQU    25H          ; DELAY FOR OPERATION COMPLETE
<1> DELAY_2      EQU    0600H        ; DELAY FOR READY
<1> DELAY_3      EQU    0100H        ; DELAY FOR DATA REQUEST
<1>
<1> HF_FAIL      EQU    08H          ; CMOS FLAG IN BYTE 0EH
<1>
<1> ;-----             COMMAND BLOCK REFERENCE
```

```
                             <1>
                             <1> ;CMD_BLOCK      EQU     BP-8            ; @CMD_BLOCK REFERENCES BLOCK HEAD IN SS
                             <1>                                 ;  (BP) POINTS TO COMMAND BLOCK TAIL
                             <1>                                 ;     AS DEFINED BY THE "ENTER" PARMS
                             <1> ; 19/12/2014
                             <1> ORG_VECTOR  equ    4*13h        ; INT 13h vector
                             <1> DISK_VECTOR equ    4*40h        ; INT 40h vector (for floppy disks)
                             <1> ;HDISK_INT  equ    4*76h        ; Primary HDC - Hardware interrupt (IRQ14)
                             <1> ;HDISK_INT1 equ    4*76h        ; Primary HDC - Hardware interrupt (IRQ14)
                             <1> ;HDISK_INT2 equ    4*77h        ; Secondary HDC - Hardware interrupt (IRQ15)
                             <1> ;HF_TBL_VEC equ    4*41h        ; Pointer to 1st fixed disk parameter table
                             <1> ;HF1_TBL_VEC equ   4*46h        ; Pointer to 2nd fixed disk parameter table
                             <1>
                             <1> align 2
                             <1>
                             <1> ;----------------------------------------------------------------
                             <1> ; FIXED DISK I/O SETUP                                         :
                             <1> ;                                                       :
                             <1> ;  -  ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK       :
                             <1> ;  -  PERFORM POWER ON DIAGNOSTICS                         :
                             <1> ;     SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED       :
                             <1> ;                                                       :
                             <1> ;----------------------------------------------------------------
                             <1>
                             <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>
                             <1> DISK_SETUP:
                             <1>       ;CLI
                             <1>       ;;MOV  AX,ABS0                   ; GET ABSOLUTE SEGMENT
                             <1>       ;xor   ax,ax
                             <1>       ;MOV   DS,AX                     ; SET SEGMENT REGISTER
                             <1>       ;MOV   AX, [ORG_VECTOR]          ; GET DISKETTE VECTOR
                             <1>       ;MOV   [DISK_VECTOR],AX          ;  INTO INT 40H
                             <1>       ;MOV   AX, [ORG_VECTOR+2]
                             <1>       ;MOV   [DISK_VECTOR+2],AX
                             <1>       ;MOV   word [ORG_VECTOR],DISK_IO ; FIXED DISK HANDLER
                             <1>       ;MOV   [ORG_VECTOR+2],CS
                             <1>       ; 1st controller (primary master, slave)   - IRQ 14
                             <1>       ;;MOV  word [HDISK_INT],HD_INT          ; FIXED DISK INTERRUPT
                             <1>       ;mov   word [HDISK_INT1],HD_INT  ;
                             <1>       ;;MOV  [HDISK_INT+2],CS
                             <1>       ;mov   [HDISK_INT1+2],CS
                             <1>       ; 2nd controller (secondary master, slave) - IRQ 15
                             <1>       ;mov   word [HDISK_INT2],HD1_INT ;
                             <1>       ;mov   [HDISK_INT2+2],CS
                             <1>       ;
                             <1>       ;;MOV  word [HF_TBL_VEC],HD0_DPT ; PARM TABLE DRIVE 80
                             <1>       ;;MOV  word [HF_TBL_VEC+2],DPT_SEGM
                             <1>       ;;MOV  word [HF1_TBL_VEC],HD1_DPT; PARM TABLE DRIVE 81
                             <1>       ;;MOV  word [HF1_TBL_VEC+2],DPT_SEGM
                             <1>       ;push  cs
                             <1>       ;pop   ds
                             <1>       ;mov   word [HDPM_TBL_VEC],HD0_DPT      ; PARM TABLE DRIVE 80h
                             <1>       ;mov   word [HDPM_TBL_VEC+2],DPT_SEGM
0000420E C705[D8580100]0000- <1>       mov    dword [HDPM_TBL_VEC], (DPT_SEGM*16)+HD0_DPT
00004216 0900                <1>
                             <1>       ;mov   word [HDPS_TBL_VEC],HD1_DPT      ; PARM TABLE DRIVE 81h
                             <1>       ;mov   word [HDPS_TBL_VEC+2],DPT_SEGM
00004218 C705[DC580100]2000- <1>       mov    dword [HDPS_TBL_VEC], (DPT_SEGM*16)+HD1_DPT
00004220 0900                <1>
                             <1>       ;mov   word [HDSM_TBL_VEC],HD2_DPT      ; PARM TABLE DRIVE 82h
                             <1>       ;mov   word [HDSM_TBL_VEC+2],DPT_SEGM
00004222 C705[E0580100]4000- <1>       mov    dword [HDSM_TBL_VEC], (DPT_SEGM*16)+HD2_DPT
0000422A 0900                <1>
                             <1>       ;mov   word [HDSS_TBL_VEC],HD3_DPT      ; PARM TABLE DRIVE 83h
                             <1>       ;mov   word [HDSS_TBL_VEC+2],DPT_SEGM
0000422C C705[E4580100]6000- <1>       mov    dword [HDSS_TBL_VEC], (DPT_SEGM*16)+HD3_DPT
00004234 0900                <1>
                             <1>       ;
                             <1>       ;;IN   AL,INTB01           ; TURN ON SECOND INTERRUPT CHIP
                             <1>       ;;;AND AL,0BFH
                             <1>       ;;and  al, 3Fh                    ; enable IRQ 14 and IRQ 15
                             <1>       ;;;JMP $+2
                             <1>       ;;IODELAY
                             <1>       ;;OUT  INTB01,AL
                             <1>       ;;IODELAY
                             <1>       ;;IN   AL,INTA01           ; LET INTERRUPTS PASS THRU TO
                             <1>       ;;AND  AL,0FBH              ;  SECOND CHIP
                             <1>       ;;;JMP $+2
                             <1>       ;;IODELAY
                             <1>       ;;OUT  INTA01,AL
                             <1>       ;
                             <1>       ;STI
                             <1>       ;;PUSH DS                   ; MOVE ABS0 POINTER TO
                             <1>       ;;POP  ES                   ; EXTRA SEGMENT POINTER
                             <1>       ;;;CALL      DDS                   ; ESTABLISH DATA SEGMENT
                             <1>       ;;MOV  byte [DISK_STATUS1],0      ; RESET THE STATUS INDICATOR
                             <1>       ;;MOV  byte [HF_NUM],0            ; ZERO NUMBER OF FIXED DISKS
                             <1>       ;;MOV  byte [CONTROL_BYTE],0
                             <1>       ;;MOV  byte [PORT_OFF],0   ; ZERO CARD OFFSET
                             <1>       ; 20/12/2014 - private code by Erdogan Tan
                             <1>                     ; (out of original PC-AT, PC-XT BIOS code)
                             <1>       ;mov   si, hd0_type
00004236 BE[F85C0000]        <1>       mov    esi, hd0_type
                             <1>       ;mov   cx, 4
0000423B B904000000          <1>       mov    ecx, 4
                             <1> hde_l:
00004240 AC                  <1>       lodsb
00004241 3C80                <1>       cmp    al, 80h                  ; 8?h = existing
00004243 7206                <1>       jb     short _L4
00004245 FE05[D4580100]      <1>       inc    byte [HF_NUM]       ; + 1 hard (fixed) disk drives
                             <1> _L4: ; 26/02/2015
0000424B E2F3                <1>       loop   hde_l
```

```
                                <1> ;_L4:                           ; 0 <= [HF_NUM] =< 4
                                <1> ;L4:
                                <1>       ;
                                <1>       ;; 31/12/2014 - cancel controller diagnostics here
                                <1>       ;;;mov       cx, 3  ; 26/12/2014 (Award BIOS 1999)
                                <1>       ;;mov  cl, 3
                                <1>       ;;
                                <1>       ;;MOV  DL,80H             ; CHECK THE CONTROLLER
                                <1> ;;hdc_dl:
                                <1>       ;;MOV  AH,14H             ; USE CONTROLLER DIAGNOSTIC COMMAND
                                <1>       ;;INT  13H                ; CALL BIOS WITH DIAGNOSTIC COMMAND
                                <1>       ;;;JC  short CTL_ERRX           ; DISPLAY ERROR MESSAGE IF BAD RETURN
                                <1>       ;;;jc  short POD_DONE ;22/12/2014
                                <1>       ;;jnc  short hdc_reset0
                                <1>       ;;loop hdc_dl
                                <1>       ;;; 27/12/2014
                                <1>       ;;stc
                                <1>       ;;retn
                                <1>       ;
                                <1> ;;hdc_reset0:
                                <1>       ; 18/01/2015
0000424D 8A0D[D4580100]         <1>       mov    cl, [HF_NUM]
00004253 20C9                   <1>       and    cl, cl
00004255 740E                   <1>       jz     short POD_DONE
                                <1>       ;
00004257 B27F                   <1>       mov    dl, 7Fh
                                <1> hdc_reset1:
00004259 FEC2                   <1>       inc    dl
                                <1>       ;; 31/12/2015
                                <1>       ;;push dx
                                <1>       ;;push cx
                                <1>       ;;push ds
                                <1>       ;;sub  ax, ax
                                <1>       ;;mov  ds, ax
                                <1>       ;;MOV  AX, [TIMER_LOW]          ; GET START TIMER COUNTS
                                <1>       ;;pop  ds
                                <1>       ;;MOV  BX,AX
                                <1>       ;;ADD  AX,6*182           ; 60 SECONDS* 18.2
                                <1>       ;;MOV  CX,AX
                                <1>       ;;mov  word [wait_count], 0      ; 22/12/2014 (reset wait counter)
                                <1>       ;;
                                <1>       ;; 31/12/2014 - cancel HD_RESET_1
                                <1>       ;;CALL HD_RESET_1         ; SET UP DRIVE 0, (1,2,3)
                                <1>       ;;pop  cx
                                <1>       ;;pop  dx
                                <1>       ;;
                                <1>       ; 18/01/2015
0000425B B40D                   <1>       mov    ah, 0Dh ; ALTERNATE RESET
                                <1>       ;int   13h
0000425D E8A4FFFFFF             <1>       call   int13h
00004262 E2F5                   <1>       loop   hdc_reset1
00004264 F8                     <1>       clc    ; 29/05/2016
                                <1> POD_DONE:
00004265 C3                     <1>       RETn
                                <1>
                                <1> ;;-----      POD_ERROR
                                <1>
                                <1> ;;CTL_ERRX:
                                <1> ;     ;MOV   SI,OFFSET F1782     ; CONTROLLER ERROR
                                <1> ;     ;CALL  SET_FAIL            ; DO NOT IPL FROM DISK
                                <1> ;     ;CALL  E_MSG               ; DISPLAY ERROR AND SET (BP) ERROR FLAG
                                <1> ;     ;JMP   short POD_DONE
                                <1>
                                <1> ;;HD_RESET_1:
                                <1> ;;    ;PUSH  BX                  ; SAVE TIMER LIMITS
                                <1> ;;    ;PUSH  CX
                                <1> ;;RES_1: MOV AH,09H              ; SET DRIVE PARAMETERS
                                <1> ;;    INT    13H
                                <1> ;;    JC     short RES_2
                                <1> ;;    MOV    AH,11H              ; RECALIBRATE DRIVE
                                <1> ;;    INT    13H
                                <1> ;;    JNC    short RES_CK        ; DRIVE OK
                                <1> ;;RES_2: ;CALL     POD_TCHK            ; CHECK TIME OUT
                                <1> ;;    cmp    word [wait_count], 6*182 ; waiting time (in timer ticks)
                                <1> ;;                              ; (30 seconds)
                                <1> ;;    ;cmc
                                <1> ;;    ;JNC   short RES_1
                                <1> ;;    jb     short RES_1
                                <1> ;;;RES_FL: ;MOV    SI,OFFSET F1781     ; INDICATE DISK 1 FAILURE;
                                <1> ;;    ;TEST  DL,1
                                <1> ;;    ;JNZ   RES_E1
                                <1> ;;    ;MOV   SI,OFFSET F1780     ; INDICATE DISK 0 FAILURE
                                <1> ;;    ;CALL  SET_FAIL            ; DO NOT TRY TO IPL DISK 0
                                <1> ;;    ;JMP   SHORT RES_E1
                                <1> ;;RES_ER: ; 22/12/2014
                                <1> ;;RES_OK:
                                <1> ;;    ;POP   CX                  ; RESTORE TIMER LIMITS
                                <1> ;;    ;POP   BX
                                <1> ;;    RETn
                                <1> ;;
                                <1> ;;RES_RS: MOV      AH,00H              ; RESET THE DRIVE
                                <1> ;;    INT    13H
                                <1> ;;RES_CK: MOV      AH,08H              ; GET MAX CYLINDER,HEAD,SECTOR
                                <1> ;;    MOV    BL,DL               ; SAVE DRIVE CODE
                                <1> ;;    INT    13H
                                <1> ;;    JC     short RES_ER
                                <1> ;;    MOV    [NEC_STATUS],CX     ; SAVE MAX CYLINDER, SECTOR
                                <1> ;;    MOV    DL,BL               ; RESTORE DRIVE CODE
                                <1> ;;RES_3: MOV AX,0401H            ; VERIFY THE LAST SECTOR
                                <1> ;;    INT    13H
                                <1> ;;    JNC    short RES_OK        ; VERIFY OK
                                <1> ;;    CMP    AH,BAD_SECTOR       ; OK ALSO IF JUST ID READ
                                <1> ;;    JE     short RES_OK
```

```
                             <1> ;;    CMP   AH,DATA_CORRECTED
                             <1> ;;    JE    short RES_OK
                             <1> ;;    CMP   AH,BAD_ECC
                             <1> ;;    JE    short RES_OK
                             <1> ;;    ;CALL POD_TCHK            ; CHECK FOR TIME OUT
                             <1> ;;    cmp   word [wait_count], 6*182 ; waiting time (in timer ticks)
                             <1> ;;                              ; (60 seconds)
                             <1> ;;    cmc
                             <1> ;;    JC    short RES_ER        ; FAILED
                             <1> ;;    MOV   CX,[NEC_STATUS]     ; GET SECTOR ADDRESS, AND CYLINDER
                             <1> ;;    MOV   AL,CL               ; SEPARATE OUT SECTOR NUMBER
                             <1> ;;    AND   AL,3FH
                             <1> ;;    DEC   AL                  ; TRY PREVIOUS ONE
                             <1> ;;    JZ    short RES_RS        ; WE'VE TRIED ALL SECTORS ON TRACK
                             <1> ;;    AND   CL,0C0H             ; KEEP CYLINDER BITS
                             <1> ;;    OR    CL,AL               ; MERGE SECTOR WITH CYLINDER BITS
                             <1> ;;    MOV   [NEC_STATUS],CX     ; SAVE CYLINDER, NEW SECTOR NUMBER
                             <1> ;;    JMP   short RES_3         ; TRY AGAIN
                             <1> ;;;RES_ER: MOV     SI,OFFSET F1791     ; INDICATE DISK 1 ERROR
                             <1> ;;    ;TEST DL,1
                             <1> ;;    ;JNZ  short RES_E1
                             <1> ;;    ;MOV  SI,OFFSET F1790     ; INDICATE DISK 0 ERROR
                             <1> ;;;RES_E1:
                             <1> ;;    ;CALL E_MSG               ; DISPLAY ERROR AND SET (BP) ERROR FLAG
                             <1> ;;;RES_OK:
                             <1> ;;    ;POP  CX                  ; RESTORE TIMER LIMITS
                             <1> ;;    ;POP  BX
                             <1> ;;    ;RETn
                             <1> ;
                             <1> ;;SET_FAIL:
                             <1> ;     ;MOV  AX,X*(CMOS_DIAG+NMI)       ; GET CMOS ERROR BYTE
                             <1> ;     ;CALL CMOS_READ
                             <1> ;     ;OR   AL,HF_FAIL          ; SET DO NOT IPL FROM DISK FLAG
                             <1> ;     ;XCHG AH,AL               ; SAVE IT
                             <1> ;     ;CALL CMOS_WRITE          ; PUT IT OUT
                             <1> ;     ;RETn
                             <1> ;
                             <1> ;;POD_TCHK:                     ; CHECK FOR 30 SECOND TIME OUT
                             <1> ;     ;POP  AX                  ; SAVE RETURN
                             <1> ;     ;POP  CX                  ; GET TIME OUT LIMITS
                             <1> ;     ;POP  BX
                             <1> ;     ;PUSH BX                  ; AND SAVE THEM AGAIN
                             <1> ;     ;PUSH CX
                             <1> ;     ;PUSH AX
                             <1> ;     ;push ds
                             <1> ;     ;xor  ax, ax
                             <1> ;     ;mov  ds, ax              ; RESTORE RETURN
                             <1> ;     ;MOV  AX, [TIMER_LOW]            ; AX = CURRENT TIME
                             <1> ;     ;                         ; BX = START TIME
                             <1> ;     ;                         ; CX = END TIME
                             <1> ;     ;pop  ds
                             <1> ;     ;CMP  BX,CX
                             <1> ;     ;JB   short TCHK1         ; START < END
                             <1> ;     ;CMP  BX,AX
                             <1> ;     ;JB   short TCHKG         ; END < START < CURRENT
                             <1> ;     ;JMP  SHORT TCHK2         ; END, CURRENT < START
                             <1> ;;TCHK1: CMP AX,BX
                             <1> ;;    JB    short TCHKNG        ; CURRENT < START < END
                             <1> ;;TCHK2: CMP AX,CX
                             <1> ;;    JB    short TCHKG         ; START < CURRENT < END
                             <1> ;;                              ; OR CURRENT < END < START
                             <1> ;;TCHKNG: STC                          ; CARRY SET INDICATES TIME OUT
                             <1> ;;    RETn
                             <1> ;;TCHKG: CLC                    ; INDICATE STILL TIME
                             <1> ;;    RETn
                             <1> ;;
                             <1> ;;int_13h:
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     FIXED DISK BIOS ENTRY POINT      :
                             <1> ;----------------------------------------
                             <1>
                             <1> ; 15/01/2017
                             <1> ; 14/01/2017
                             <1> ; 07/01/2017
                             <1> ; 02/01/2017
                             <1> ; 01/06/2016
                             <1> ; 16/05/2016, 27/05/2016, 28/05/2016, 29/05/2016
                             <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> int33h:  ; DISK I/O
                             <1>       ; 29/05/2016
00004266 80642408FE          <1>       and   byte [esp+8], 11111110b  ; clear carry bit of eflags register
                             <1>       ; 16/05/2016
0000426B 1E                  <1>       push  ds
0000426C 53                  <1>       push  ebx ; user's buffer address (virtual)
0000426D 66BB1000            <1>       mov   bx, KDATA ; System (Kernel's) data segment
00004271 8EDB                <1>       mov   ds, bx
                             <1>
                             <1>       ;;15/01/2017
                             <1>       ; 14/01/2017
                             <1>       ; 02/01/2017
                             <1>       ;;mov byte [intflg], 33h  ; disk io interrupt
                             <1>       ;pop  ebx
                             <1>       ;mov  [user_buffer], ebx
                             <1>
00004273 8F05[C8650100]      <1>       pop   dword [user_buffer] ; 01/06/2016
                             <1>
00004279 C605[FE5E0100]00    <1>       mov   byte [scount], 0 ; sector count for transfer
00004280 80FC03              <1>       cmp   ah, 03h ; chs write
00004283 7744                <1>       ja    short int33h_2
00004285 7407                <1>       je    short int33h_0
00004287 80FC02              <1>       cmp   ah, 02h ; chs read
0000428A 726A                <1>       jb    short int33h_5
```

```
0000428C EB63                 <1>       jmp    short int33h_4
                              <1> int33h_0:
                              <1>       ; transfer user's buffer content to sector buffer
0000428E 51                   <1>       push   ecx
0000428F 0FB6C8               <1>       movzx  ecx, al
                              <1> int33h_1:
00004292 56                   <1>       push   esi
00004293 8B35[C8650100]       <1>       mov    esi, [user_buffer]
                              <1>       ; esi = user's buffer address (virtual, ebx)
00004299 57                   <1>       push   edi
0000429A 06                   <1>       push   es
0000429B 50                   <1>       push   eax
0000429C 66B81000             <1>       mov    ax, KDATA
000042A0 8EC0                 <1>       mov    es, ax
000042A2 BF00000700           <1>       mov    edi, Cluster_Buffer
000042A7 C1E109               <1>       shl    ecx, 9 ; * 512
000042AA E814A50000           <1>       call   transfer_from_user_buffer
000042AF 58                   <1>       pop    eax
000042B0 07                   <1>       pop    es
000042B1 5F                   <1>       pop    edi
000042B2 5E                   <1>       pop    esi
000042B3 59                   <1>       pop    ecx
000042B4 7340                 <1>       jnc    short int33h_5
000042B6 8B1D[C8650100]       <1>       mov    ebx, [user_buffer] ; 01/06/2016
000042BC 1F                   <1>       pop    ds
                              <1>
                              <1>       ;;15/01/2017
                              <1>       ; 02/01/2017
                              <1>       ;cli
                              <1>       ;;mov  byte [ss:intflg], 0 ; 07/01/2017
                              <1>       ;
                              <1>       ; (*) 29/05/2016
                              <1>       ; (*) retf 4 ; skip eflags on stack
                              <1>
                              <1>       ; 29/05/2016 -set carry flag on stack-
                              <1>       ; [esp] = EIP
                              <1>       ; [esp+4] = CS
                              <1>       ; [esp+8] = E-FLAGS
000042BD 804C240801           <1>       or     byte [esp+8], 1  ; set carry bit of eflags register
                              <1>       ; [esp+12] = ESP (user)
                              <1>       ; [esp+16] = SS (User)
000042C2 B8FF000000           <1>       mov    eax, 0FFh ; Unknown error !?
                              <1>       ;iretd
000042C7 EB79                 <1>       jmp    short int33h_7  ; 07/01/2017
                              <1>
                              <1>       ; (*) 29/05/2016 - 'ref 4' intruction causes to stack fault
                              <1>       ; (OUTER-PRIVILEGE-LEVEL)
                              <1>       ; INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986
                              <1>       ; // RETF instruction:
                              <1>       ;
                              <1>       ; IF OperandMode=32 THEN
                              <1>       ;    Load CS:EIP from stack;
                              <1>       ;    Set CS RPL to CPL;
                              <1>       ;    Increment eSP by 8 plus the immediate offset if it exists;
                              <1>       ;    Load SS:eSP from stack;
                              <1>       ; ELSE (* OperandMode=16 *)
                              <1>       ;    Load CS:IP from stack;
                              <1>       ;    Set CS RPL to CPL;
                              <1>       ;    Increment eSP by 4 plus the immediate offset if it exists;
                              <1>       ;    Load SS:eSP from stack;
                              <1>       ; FI;
                              <1>       ;
                              <1>       ; //
                              <1>
                              <1> int33h_2:
000042C9 80FC05               <1>       cmp    ah, 05h ; format track
000042CC 770A                 <1>       ja     short int33h_3
000042CE 7226                 <1>       jb     short int33h_5
000042D0 51                   <1>       push   ecx
000042D1 B901000000           <1>       mov    ecx, 1
000042D6 EBBA                 <1>       jmp    short int33h_1
                              <1> int33h_3:
000042D8 80FC1C               <1>       cmp    ah, 1Ch ; LBA write
000042DB 7719                 <1>       ja     short int33h_5
000042DD 74AF                 <1>       je     short int33h_0
000042DF 80FC1B               <1>       cmp    ah, 1Bh ; LBA read
000042E2 740D                 <1>       je     short int33h_4
000042E4 80FC08               <1>       cmp    ah, 08h ; get disk parameters
000042E7 750D                 <1>       jne    short int33h_5
                              <1>       ; 01/06/2016
000042E9 8B1D[C8650100]       <1>       mov    ebx, [user_buffer] ; user's buffer address
000042EF EB0A                 <1>       jmp    short int33h_6
                              <1> int33h_4:
000042F1 A2[FE5E0100]         <1>       mov    byte [scount], al ; <= 128 sectors
                              <1> int33h_5:
000042F6 BB00000700           <1>       mov    ebx, Cluster_Buffer ; max. 65536 bytes
                              <1>                               ; buf. addr: 70000h
                              <1>       ;mov   byte [ClusterBuffer_Valid], 0
                              <1> int33h_6:
000042FB 1F                   <1>       pop    ds
000042FC 9C                   <1>       pushfd
000042FD 0E                   <1>       push   cs
000042FE E84D000000           <1>       call   DISK_IO
00004303 2E8B1D[C8650100]     <1>       mov    ebx, [CS:user_buffer] ; 01/06/2016
0000430A 723D                 <1>       jc     short int33h_9
                              <1>       ;
0000430C 2E803D[FE5E0100]00   <1>       cmp    byte [CS:scount], 0
00004314 762C                 <1>       jna    short int33h_7
                              <1>       ; transfer sector buffer content to user's buffer
00004316 06                   <1>       push   es
00004317 1E                   <1>       push   ds
00004318 50                   <1>       push   eax
00004319 66B81000             <1>       mov    ax, KDATA
```

```
0000431D 8ED8                <1>       mov    ds, ax
0000431F 8EC0                <1>       mov    es, ax
00004321 51                  <1>       push   ecx
00004322 56                  <1>       push   esi
00004323 57                  <1>       push   edi
00004324 0FB60D[FE5E0100]    <1>       movzx ecx, byte [scount]
0000432B C1E109              <1>       shl    ecx, 9 ; * 512 bytes
0000432E 89DF                <1>       mov    edi, ebx ; user's buffer address
00004330 BE00000700          <1>       mov    esi, Cluster_Buffer
00004335 E83FA40000          <1>       call   transfer_to_user_buffer
0000433A 5F                  <1>       pop    edi
0000433B 5E                  <1>       pop    esi
0000433C 59                  <1>       pop    ecx
0000433D 58                  <1>       pop    eax
0000433E 1F                  <1>       pop    ds
0000433F 07                  <1>       pop    es
00004340 7202                <1>       jc     short int33h_8
                             <1> int33h_7:
00004342 FA                  <1>       cli
                             <1>       ;;15/01/2017
                             <1>       ;;mov  byte [ss:intflg], 0 ; 07/01/2017
                             <1>       ; cf = 0  ; use eflags which is in stack
00004343 CF                  <1>       iretd
                             <1> int33h_8:
00004344 B8FF000000          <1>       mov    eax, 0FFh ; Unknown error !?
                             <1> int33h_9:
                             <1>       ; cf = 1
                             <1>
                             <1>       ; (*) 29/05/2016
                             <1>       ; (*) retf 4 ; skip eflags on stack
                             <1>       ; Note: This 'retf 4' was wrong, -it was causing
                             <1>       ;       to stack errors in ring 3-
                             <1>       ;      POP sequence of 'retf 4' is as
                             <1>       ;       "eip, cs, eflags, esp, ss, +4 bytes"
                             <1>         ;       it is not as "eip, cs, +4 bytes, esp, ss" !
                             <1>
                             <1>       ; 29/05/2016 -set carry flag on stack-
00004349 804C240801          <1>       or     byte [esp+8], 1  ; set carry bit of eflags register
                             <1>       ;iretd
0000434E EBF2                <1>       jmp    short int33h_7 ; 07/01/2017
                             <1>
                             <1> ; 09/12/2017
                             <1> ; 29/05/2016
                             <1> ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>
                             <1> DISK_IO:
00004350 80FA80              <1>       CMP    DL,80H              ; TEST FOR FIXED DISK DRIVE
                             <1>       ;JAE   short A1            ; YES, HANDLE HERE
                             <1>       ;;;INT 40H                 ; DISKETTE HANDLER
                             <1>       ;;call int40h
00004353 0F8222F0FFFF        <1>       jb     DISKETTE_IO_1
                             <1> ;RET_2:
                             <1>       ;RETF  2                   ; BACK TO CALLER
                             <1> ;     retf   4
                             <1> A1:
00004359 FB                  <1>       STI                        ; ENABLE INTERRUPTS
                             <1>       ;; 04/01/2015
                             <1>       ;;OR   AH,AH
                             <1>       ;;JNZ  short A2
                             <1>       ;;INT  40H                 ; RESET NEC WHEN AH=0
                             <1>       ;;SUB  AH,AH
0000435A 80FA83              <1>       CMP    DL,(80H + S_MAX_FILE - 1)
                             <1>       ;JA    short RET_2
0000435D 7616                <1>       jna    short _A0
                             <1>       ; 29/05/2016
0000435F 1E                  <1>       push   ds
00004360 6650                <1>       push   ax
00004362 66B81000            <1>       mov    ax, KDATA
00004366 8ED8                <1>       mov    ds, ax
00004368 6658                <1>       pop    ax
0000436A B4AA                <1>         mov     ah, 0AAh         ; Hard disk drive not ready !
                             <1>                            ; (Programmer's guide to AMIBIOS, 1992)
0000436C 8825[D3580100]      <1>       mov     byte [DISK_STATUS1], ah
00004372 1F                  <1>       pop    ds
00004373 EB38                <1>       jmp    short RET_2
                             <1> _A0:
                             <1>       ; 18/01/2015
00004375 08E4                <1>       or     ah,ah
00004377 743A                <1>       jz     short A4
00004379 80FC0D              <1>       cmp    ah, 0Dh      ; Alternate reset
0000437C 7504                <1>       jne    short A2
0000437E 28E4                <1>       sub    ah,ah  ; Reset
00004380 EB31                <1>       jmp    short A4
                             <1> A2:
00004382 80FC08              <1>       CMP    AH,08H              ; GET PARAMETERS IS A SPECIAL CASE
                             <1>       ;JNZ   short A3
                             <1>         ;JMP     GET_PARM_N
00004385 0F8432030000        <1>       je     GET_PARM_N
0000438B 80FC15              <1> A3:   CMP    AH,15H              ; READ DASD TYPE IS ALSO
                             <1>       ;JNZ   short A4
                             <1>         ;JMP     READ_DASD_TYPE
0000438E 0F84DB020000        <1>         je      READ_DASD_TYPE
                             <1>       ; 02/02/2015
00004394 80FC1D              <1>       cmp    ah, 1Dh                     ;(Temporary for Retro UNIX 386 v1)
                             <1>       ; 12/01/2015
00004397 F5                  <1>       cmc
00004398 7319                <1>       jnc    short A4
                             <1> int33h_bad_cmd:
                             <1>       ; 16/05/2016
                             <1>       ; 30/01/2015
                             <1>       ; 29/05/2016
0000439A 1E                  <1>       push   ds
0000439B 6650                <1>       push   ax
```

```
0000439D 66B81000            <1>       mov    ax, KDATA
000043A1 8ED8                <1>       mov    ds, ax
000043A3 6658                <1>       pop    ax
000043A5 B401                <1>       mov    ah, BAD_CMD
000043A7 8825[D3580100]      <1>       mov     [DISK_STATUS1], ah ; BAD_CMD  ; COMMAND ERROR
                             <1>         ;jmp short RET_2
                             <1> RET_2:
                             <1>       ; (*) 29/05/2016
                             <1>       ; (*) retf 4
000043AD 804C240801          <1>       or     byte [esp+8], 1 ; set carry bit of eflags register
000043B2 CF                  <1>       iretd
                             <1> A4:                             ; SAVE REGISTERS DURING OPERATION
000043B3 C8080000            <1>       ENTER  8,0                ; SAVE (BP) AND MAKE ROOM FOR @CMD_BLOCK
000043B7 53                  <1>       PUSH   eBX                ;  IN THE STACK, THE COMMAND BLOCK IS:
000043B8 51                  <1>       PUSH   eCX                ;   @CMD_BLOCK == BYTE PTR [BP]-8
000043B9 52                  <1>       PUSH   eDX
000043BA 1E                  <1>       PUSH   DS
000043BB 06                  <1>       PUSH   ES
000043BC 56                  <1>       PUSH   eSI
000043BD 57                  <1>       PUSH   eDI
                             <1>       ;;04/01/2015
                             <1>       ;;OR   AH,AH              ; CHECK FOR RESET
                             <1>       ;;JNZ  short A5
                             <1>       ;;MOV  DL,80H             ; FORCE DRIVE 80 FOR RESET
                             <1> ;;A5:
                             <1>       ;push  cs
                             <1>       ;pop   ds
                             <1>       ; 21/02/2015
000043BE 6650                <1>       push   ax
000043C0 66B81000            <1>       mov    ax, KDATA
000043C4 8ED8                <1>       mov    ds, ax
000043C6 8EC0                <1>       mov    es, ax
000043C8 6658                <1>       pop    ax
000043CA E88D000000          <1>       CALL   DISK_IO_CONT       ; PERFORM THE OPERATION
                             <1>       ;;CALL DDS                ; ESTABLISH SEGMENT
000043CF 8A25[D3580100]      <1>       MOV    AH,[DISK_STATUS1]  ; GET STATUS FROM OPERATION
                             <1>       ;(*) CMP AH,1             ; SET THE CARRY FLAG TO INDICATE
                             <1>       ;(*) CMC                  ; SUCCESS OR FAILURE
000043D5 5F                  <1>       POP    eDI                ; RESTORE REGISTERS
000043D6 5E                  <1>       POP    eSI
000043D7 07                  <1>         POP     ES
000043D8 1F                  <1>         POP     DS
000043D9 5A                  <1>       POP    eDX
000043DA 59                  <1>       POP    eCX
000043DB 5B                  <1>       POP    eBX
000043DC C9                  <1>       LEAVE                     ; ADJUST (SP) AND RESTORE (BP)
                             <1>       ;RETF  2                  ; THROW AWAY SAVED FLAGS
                             <1>       ; (*) 29/05/2016
                             <1>       ; (*) retf 4
000043DD 80FC01              <1>       cmp    ah, 1
000043E0 7205                <1>       jc     short _A5
000043E2 804C240801          <1>       or     byte [esp+8], 1 ; set carry bit of eflags register
                             <1> _A5:
000043E7 CF                  <1>       iretd
                             <1>
                             <1> ; 21/02/2015
                             <1> ;       dw --> dd
                             <1> D1:                             ; FUNCTION TRANSFER TABLE
000043E8 [AB450000]          <1>       dd     DISK_RESET         ; 000H
000043EC [22460000]          <1>       dd     RETURN_STATUS      ; 001H
000043F0 [2F460000]          <1>       dd     DISK_READ          ; 002H
000043F4 [38460000]          <1>       dd     DISK_WRITE         ; 003H
000043F8 [41460000]          <1>       dd     DISK_VERF          ; 004H
000043FC [59460000]          <1>       dd     FMT_TRK            ; 005H
00004400 [A1450000]          <1>       dd     BAD_COMMAND        ; 006H FORMAT BAD SECTORS
00004404 [A1450000]          <1>       dd     BAD_COMMAND        ; 007H FORMAT DRIVE
00004408 [A1450000]          <1>       dd     BAD_COMMAND        ; 008H RETURN PARAMETERS
0000440C [44470000]          <1>       dd     INIT_DRV           ; 009H
00004410 [A3470000]          <1>       dd     RD_LONG            ; 00AH
00004414 [AC470000]          <1>       dd     WR_LONG            ; 00BH
00004418 [B5470000]          <1>       dd     DISK_SEEK          ; 00CH
0000441C [AB450000]          <1>       dd     DISK_RESET         ; 00DH
00004420 [A1450000]          <1>       dd     BAD_COMMAND        ; 00EH READ BUFFER
00004424 [A1450000]          <1>       dd     BAD_COMMAND        ; 00FH WRITE BUFFER
00004428 [DD470000]          <1>       dd     TST_RDY            ; 010H
0000442C [01480000]          <1>       dd     HDISK_RECAL        ; 011H
00004430 [A1450000]          <1>       dd     BAD_COMMAND        ; 012H MEMORY DIAGNOSTIC
00004434 [A1450000]          <1>       dd     BAD_COMMAND        ; 013H DRIVE DIAGNOSTIC
00004438 [37480000]          <1>       dd     CTLR_DIAGNOSTIC    ; 014H CONTROLLER DIAGNOSTIC
                             <1>       ; 02/02/2015 (Temporary - Retro UNIX 386 v1 - DISK I/O test)
0000443C [A1450000]          <1>       dd     BAD_COMMAND        ; 015h
00004440 [A1450000]          <1>       dd     BAD_COMMAND        ; 016h
00004444 [A1450000]          <1>       dd     BAD_COMMAND        ; 017h
00004448 [A1450000]          <1>       dd     BAD_COMMAND        ; 018h
0000444C [A1450000]          <1>       dd     BAD_COMMAND        ; 019h
00004450 [A1450000]          <1>       dd     BAD_COMMAND        ; 01Ah
00004454 [2F460000]          <1>       dd     DISK_READ          ; 01Bh ; LBA read
00004458 [38460000]          <1>       dd     DISK_WRITE         ; 01Ch ; LBA write
                             <1> D1L     EQU    $ - D1
                             <1>
                             <1> DISK_IO_CONT:
                             <1>       ;;CALL DDS                ; ESTABLISH SEGMENT
0000445C 80FC01              <1>       CMP    AH,01H             ; RETURN STATUS
                             <1>       ;;JNZ  short SU0
                             <1>         ;;JMP    RETURN_STATUS
0000445F 0F84BD010000        <1>       je     RETURN_STATUS
                             <1> SU0:
00004465 C605[D3580100]00    <1>       MOV    byte [DISK_STATUS1],0     ; RESET THE STATUS INDICATOR
                             <1>       ;;PUSH BX                 ; SAVE DATA ADDRESS
                             <1>       ;mov   si, bx ;; 14/02/2015
0000446C 89DE                <1>       mov    esi, ebx ; 21/02/2015
0000446E 8A1D[D4580100]      <1>       MOV    BL,[HF_NUM]        ; GET NUMBER OF DRIVES
                             <1>       ;; 04/01/2015
                             <1>       ;;PUSH AX
00004474 80E27F              <1>       AND    DL,7FH             ; GET DRIVE AS 0 OR 1
                             <1>                                 ; (get drive number as 0 to 3)
00004477 38D3                <1>       CMP    BL,DL
                             <1>         ;;JBE   BAD_COMMAND_POP         ; INVALID DRIVE
00004479 0F8622010000        <1>         jbe     BAD_COMMAND ;; 14/02/2015
                             <1>         ;
                             <1>       ;;03/01/2015
0000447F 29DB                <1>       sub    ebx, ebx
00004481 88D3                <1>       mov    bl, dl
                             <1>       ;sub   bh, bh
00004483 883D[E8580100]      <1>       mov    [LBAMode], bh      ; 0
                             <1>       ;;test byte [bx+hd0_type], 1     ; LBA ready ?
                             <1>       ;test  byte [ebx+hd0_type], 1
                             <1>       ;jz    short su1          ; no
                             <1>       ;inc   byte [LBAMode]
                             <1> ;su1:
                             <1>       ; 21/02/2015 (32 bit modification)
                             <1>       ;04/01/2015
00004489 6650                <1>       push   ax ; ***
                             <1>       ;PUSH  ES ; **
0000448B 6652                <1>       PUSH   DX ; *
0000448D 6650                <1>       push   ax
0000448F E889060000          <1>       CALL   GET_VEC            ; GET DISK PARAMETERS
                             <1>       ; 02/02/2015
                             <1>       ;mov   ax, [ES:BX+16] ; I/O port base address (1F0h, 170h)
00004494 668B4310            <1>       mov    ax, [ebx+16]
00004498 66A3[E85C0000]      <1>       mov    [HF_PORT], ax
                             <1>       ;mov   dx, [ES:BX+18] ; control port address (3F6h, 376h)
0000449E 668B5312            <1>       mov    dx, [ebx+18]
000044A2 668915[EA5C0000]    <1>       mov    [HF_REG_PORT], dx
                             <1>       ;mov   al, [ES:BX+20] ; head register upper nibble (A0h,B0h,E0h,F0h)
000044A9 8A4314              <1>       mov    al, [ebx+20]
                             <1>       ; 23/02/2015
000044AC A840                <1>       test   al, 40h       ; LBA bit (bit 6)
000044AE 7406                <1>       jz     short su1
000044B0 FE05[E8580100]      <1>       inc    byte [LBAMode] ; 1
                             <1> su1:
000044B6 C0E804              <1>       shr    al, 4
000044B9 2401                <1>       and    al, 1
000044BB A2[EC5C0000]        <1>       mov    [hf_m_s], al
                             <1>       ;
                             <1>       ; 03/01/2015
                             <1>       ;MOV   AL,byte [ES:BX+8]  ; GET CONTROL BYTE MODIFIER
000044C0 8A4308              <1>       mov    al, [ebx+8]
                             <1>       ;MOV   DX,[HF_REG_PORT]   ; Device Control register
000044C3 EE                  <1>       OUT    DX,AL              ; SET EXTRA HEAD OPTION
                             <1>                                 ; Control Byte:  (= 08h, here)
                             <1>                                 ; bit 0 - 0
                             <1>                                 ; bit 1 - nIEN (1 = disable irq)
                             <1>                                 ; bit 2 - SRST (software RESET)
                             <1>                                 ; bit 3 - use extra heads (8 to 15)
                             <1>                                 ;         -always set to 1-
                             <1>                                 ; (bits 3 to 7 are reserved
                             <1>                                 ;          for ATA devices)
000044C4 8A25[D5580100]      <1>       MOV    AH,[CONTROL_BYTE]  ; SET EXTRA HEAD OPTION IN
000044CA 80E4C0              <1>       AND    AH,0C0H            ; CONTROL BYTE
000044CD 08C4                <1>       OR     AH,AL
000044CF 8825[D5580100]      <1>       MOV    [CONTROL_BYTE],AH
                             <1>       ; 04/01/2015
000044D5 6658                <1>       pop    ax
000044D7 665A                <1>       pop    dx ; * ;; 14/02/2015
000044D9 20E4                <1>       and    ah, ah ; Reset function ?
000044DB 7507                <1>       jnz    short su2
                             <1>       ;;pop  dx ; * ;; 14/02/2015
                             <1>       ;pop   es ; **
000044DD 6658                <1>       pop    ax ; ***
                             <1>       ;;pop  bx
000044DF E9C7000000          <1>         jmp     DISK_RESET
                             <1> su2:
000044E4 803D[E8580100]00    <1>       cmp    byte [LBAMode], 0
000044EB 7662                <1>       jna    short su3
                             <1>       ;
                             <1>       ; 02/02/2015 (LBA read/write function calls)
000044ED 80FC1B              <1>       cmp    ah, 1Bh
000044F0 720B                <1>       jb     short lbarw1
000044F2 80FC1C              <1>       cmp    ah, 1Ch
000044F5 775D                <1>       ja     short invldfnc
                             <1>       ;;pop  dx ; * ; 14/02/2015
                             <1>       ;mov   ax, cx ; Lower word of LBA address (bits 0-15)
000044F7 89C8                <1>       mov    eax, ecx ; LBA address (21/02/2015)
                             <1>       ;; 14/02/2015
000044F9 88D1                <1>       mov    cl, dl ; 14/02/2015
                             <1>       ;;mov  dx, bx
                             <1>       ;mov   dx, si ; higher word of LBA address (bits 16-23)
                             <1>       ;;mov  bx, di
                             <1>       ;mov   si, di ; Buffer offset
000044FB EB32                <1>       jmp    short lbarw2
                             <1> lbarw1:
                             <1>       ; convert CHS to LBA
                             <1>       ;
                             <1>       ; LBA calculation - AWARD BIOS - 1999 - AHDSK.ASM
                             <1>       ; LBA = "# of Heads" * Sectors/Track * Cylinder + Head * Sectors/Track
                             <1>       ;      + Sector - 1
000044FD 6652                <1>       push   dx ; * ;; 14/02/2015
                             <1>       ;xor   dh, dh
000044FF 31D2                <1>       xor    edx, edx
                             <1>       ;mov   dl, [ES:BX+14]     ; sectors per track (logical)
00004501 8A530E              <1>       mov    dl, [ebx+14]
                             <1>       ;xor   ah, ah
00004504 31C0                <1>       xor    eax, eax
                             <1>       ;mov   al, [ES:BX+2]; heads (logical)
00004506 8A4302              <1>       mov    al, [ebx+2]
```

```
00004509 FEC8                <1>       dec    al
0000450B 6640                <1>       inc    ax          ; 0 =  256
0000450D 66F7E2              <1>       mul    dx
                             <1>             ; AX = # of Heads" * Sectors/Track
00004510 6689CA              <1>       mov    dx, cx
                             <1>       ;and   cx, 3Fh       ; sector  (1 to 63)
00004513 83E13F              <1>       and    ecx, 3fh
00004516 86D6                <1>       xchg   dl, dh
00004518 C0EE06              <1>       shr    dh, 6
                             <1>             ; DX = cylinder (0 to 1023)
                             <1>       ;mul   dx
                             <1>             ; DX:AX = # of Heads" * Sectors/Track * Cylinder
0000451B F7E2                <1>       mul    edx
0000451D FEC9                <1>       dec    cl  ; sector - 1
                             <1>       ;add   ax, cx
                             <1>       ;adc   dx, 0
                             <1>             ; DX:AX = # of Heads" * Sectors/Track * Cylinder + Sector -1
0000451F 01C8                <1>       add    eax, ecx
00004521 6659                <1>       pop    cx ; * ; ch = head, cl = drive number (zero based)
                             <1>       ;push  dx
                             <1>       ;push  ax
00004523 50                  <1>       push   eax
                             <1>       ;mov   al, [ES:BX+14]      ; sectors per track (logical)
00004524 8A430E              <1>       mov    al, [ebx+14]
00004527 F6E5                <1>       mul    ch
                             <1>             ;  AX = Head * Sectors/Track
00004529 0FB7C0              <1>         movzx      eax, ax ; 09/12/2017
                             <1>       ;pop   dx
0000452C 5A                  <1>       pop    edx
                             <1>       ;add   ax, dx
                             <1>       ;pop   dx
                             <1>       ;adc   dx, 0 ; add carry bit
0000452D 01D0                <1>       add    eax, edx
                             <1> lbarw2:
0000452F 29D2                <1>       sub    edx, edx ; 21/02/2015
00004531 88CA                <1>       mov    dl, cl ; 21/02/2015
00004533 C645F800            <1>         mov     byte [CMD_BLOCK], 0 ; Features Register
                             <1>                          ; NOTE: Features register (1F1h, 171h)
                             <1>                          ; is not used for ATA device R/W functions.
                             <1>                          ; It is old/obsolete 'write precompensation'
                             <1>                          ; register and error register
                             <1>                          ; for old ATA/IDE devices.
                             <1>       ; 18/01/2014
                             <1>       ;mov   ch, [hf_m_s] ; Drive 0 (master) or 1 (slave)
00004537 8A0D[EC5C0000]      <1>       mov    cl, [hf_m_s]
                             <1>       ;shl   ch, 4       ; bit 4 (drive bit)
                             <1>       ;or    ch, 0E0h     ; bit 5 = 1
                             <1>                          ; bit 6 = 1 = LBA mode
                             <1>                          ; bit 7 = 1
0000453D 80C90E              <1>       or     cl, 0Eh ; 1110b
                             <1>       ;and   dh, 0Fh             ; LBA byte 4 (bits 24 to 27)
00004540 25FFFFFF0F          <1>       and    eax, 0FFFFFFFh
00004545 C1E11C              <1>       shl    ecx, 28 ; 21/02/2015
                             <1>       ;or    dh, ch
00004548 09C8                <1>       or     eax, ecx
                             <1>       ;;mov  [CMD_BLOCK+2], al ; LBA byte 1 (bits 0 to 7)
                             <1>                            ; (Sector Number Register)
                             <1>       ;;mov  [CMD_BLOCK+3], ah ; LBA byte 2 (bits 8 to 15)
                             <1>                            ; (Cylinder Low Register)
                             <1>       ;mov   [CMD_BLOCK+2], ax ; LBA byte 1, 2
                             <1>       ;mov   [CMD_BLOCK+4], dl ; LBA byte 3 (bits 16 to 23)
                             <1>                            ; (Cylinder High Register)
                             <1>       ;;mov  [CMD_BLOCK+5], dh ; LBA byte 4 (bits 24 to 27)
                             <1>                            ; (Drive/Head Register)
                             <1>
                             <1>       ;mov   [CMD_BLOCK+4], dx ; LBA byte 4, LBA & DEV select bits
0000454A 8945FA              <1>       mov    [CMD_BLOCK+2], eax ; 21/02/2015
                             <1>       ;14/02/2015
                             <1>       ;mov   dl, cl ; Drive number (INIT_DRV)
0000454D EB38                <1>       jmp    short su4
                             <1> su3:
                             <1>       ; 02/02/2015
                             <1>       ; (Temporary functions 1Bh & 1Ch are not valid for CHS mode)
0000454F 80FC14              <1>       cmp    ah, 14h
00004552 7604                <1>       jna    short chsfnc
                             <1> invldfnc:
                             <1>         ; 14/02/2015
                             <1>       ;pop   es ; **
00004554 6658                <1>         pop     ax ; ***
                             <1>         ;jmp     short BAD_COMMAND_POP
00004556 EB49                <1>         jmp     short BAD_COMMAND
                             <1> chsfnc:
                             <1>       ;MOV   AX,[ES:BX+5]        ; GET WRITE PRE-COMPENSATION CYLINDER
00004558 668B4305            <1>       mov    ax, [ebx+5]
0000455C 66C1E802            <1>       SHR    AX,2
00004560 8845F8              <1>       MOV    [CMD_BLOCK],AL
                             <1>       ;;MOV  AL,[ES:BX+8]        ; GET CONTROL BYTE MODIFIER
                             <1>       ;;PUSH DX
                             <1>       ;;MOV  DX,[HF_REG_PORT]
                             <1>       ;;OUT  DX,AL               ; SET EXTRA HEAD OPTION
                             <1>       ;;POP  DX ; *
                             <1>       ;;POP  ES ; **
                             <1>       ;;MOV  AH,[CONTROL_BYTE]   ; SET EXTRA HEAD OPTION IN
                             <1>       ;;AND  AH,0C0H             ; CONTROL BYTE
                             <1>       ;;OR   AH,AL
                             <1>       ;;MOV  [CONTROL_BYTE],AH
                             <1>       ;
00004563 88C8                <1>       MOV    AL,CL               ; GET SECTOR NUMBER
00004565 243F                <1>       AND    AL,3FH
00004567 8845FA              <1>       MOV    [CMD_BLOCK+2],AL
0000456A 886DFB              <1>       MOV    [CMD_BLOCK+3],CH    ; GET CYLINDER NUMBER
0000456D 88C8                <1>       MOV    AL,CL
0000456F C0E806              <1>       SHR    AL,6
```

```
00004572 8845FC              <1>       MOV    [CMD_BLOCK+4],AL    ; CYLINDER HIGH ORDER 2 BITS
                             <1>       ;;05/01/2015
                             <1>       ;;MOV  AL,DL               ; DRIVE NUMBER
00004575 A0[EC5C0000]        <1>       mov    al, [hf_m_s]
0000457A C0E004              <1>       SHL    AL,4
0000457D 80E60F              <1>       AND    DH,0FH              ; HEAD NUMBER
00004580 08F0                <1>       OR     AL,DH
                             <1>       ;OR    AL,80H or 20H
00004582 0CA0                <1>       OR     AL,80h+20h          ; ECC AND 512 BYTE SECTORS
00004584 8845FD              <1>       MOV    [CMD_BLOCK+5],AL    ; ECC/SIZE/DRIVE/HEAD
                             <1> su4:
                             <1>       ;POP   ES ; **
                             <1>         ;; 14/02/2015
                             <1>         ;;POP   AX
                             <1>         ;;MOV   [CMD_BLOCK+1],AL        ; SECTOR COUNT
                             <1>         ;;PUSH  AX
                             <1>         ;;MOV   AL,AH                   ; GET INTO LOW BYTE
                             <1>         ;;XOR   AH,AH                   ; ZERO HIGH BYTE
                             <1>         ;;SAL   AX,1                    ; *2 FOR TABLE LOOKUP
00004587 6658                <1>         pop     ax ; ***
00004589 8845F9              <1>         mov     [CMD_BLOCK+1], al
0000458C 29DB                <1>         sub  ebx, ebx
0000458E 88E3                <1>       mov     bl, ah
                             <1>         ;xor     bh, bh
                             <1>         ;sal     bx, 1
00004590 66C1E302            <1>         sal  bx, 2  ; 32 bit offset (21/02/2015)
                             <1>       ;;MOV   SI,AX                   ; PUT INTO SI FOR BRANCH
                             <1>         ;;CMP   AX,D1L                  ; TEST WITHIN RANGE
                             <1>         ;;JNB   short BAD_COMMAND_POP
                             <1>         ;cmp     bx, D1L
00004594 83FB74              <1>       cmp    ebx, D1L
00004597 7308                <1>       jnb    short BAD_COMMAND
                             <1>         ;xchg    bx, si
00004599 87DE                <1>         xchg ebx, esi
                             <1>       ;;;POP AX                   ; RESTORE AX
                             <1>       ;;;POP BX                   ; AND DATA ADDRESS
                             <1>
                             <1>       ;;PUSH CX
                             <1>       ;;PUSH AX                   ; ADJUST ES:BX
                             <1>       ;MOV   CX,BX                ; GET 3 HIGH ORDER NIBBLES OF BX
                             <1>       ;SHR   CX,4
                             <1>       ;MOV   AX,ES
                             <1>       ;ADD   AX,CX
                             <1>       ;MOV   ES,AX
                             <1>       ;AND   BX,000FH             ; ES:BX CHANGED TO ES:000X
                             <1>       ;;POP  AX
                             <1>       ;;POP  CX
                             <1>       ;;JMP  word [CS:SI+D1]
                             <1>       ;jmp   word [SI+D1]
0000459B FFA6[E8430000]      <1>       jmp    dword [esi+D1]
                             <1> ;;BAD_COMMAND_POP:
                             <1> ;;    POP    AX
                             <1> ;;    POP    BX
                             <1> BAD_COMMAND:
000045A1 C605[D3580100]01    <1>         MOV     byte [DISK_STATUS1],BAD_CMD  ; COMMAND ERROR
000045A8 B000                <1>       MOV    AL,0
000045AA C3                  <1>       RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     RESET THE DISK SYSTEM  (AH=00H) :
                             <1> ;----------------------------------------
                             <1>
                             <1> ; 18-1-2015 : one controller reset (not other one)
                             <1>
                             <1> DISK_RESET:
000045AB FA                  <1>       CLI
000045AC E4A1                <1>       IN     AL,INTB01           ; GET THE MASK REGISTER
                             <1>       ;JMP   $+2
                             <1>       IODELAY
000045AE EB00                <2>  jmp short $+2
000045B0 EB00                <2>  jmp short $+2
                             <1>       ;AND   AL,0BFH             ; ENABLE FIXED DISK INTERRUPT
000045B2 243F                <1>       and    al,3Fh              ; 22/12/2014 (IRQ 14 & IRQ 15)
000045B4 E6A1                <1>       OUT    INTB01,AL
000045B6 FB                  <1>       STI                        ; START INTERRUPTS
                             <1>       ; 14/02/2015
000045B7 6689D7              <1>       mov    di, dx
                             <1>       ; 04/01/2015
                             <1>       ;xor   di,di
                             <1> drst0:
000045BA B004                <1>       MOV    AL,04H  ; bit 2 - SRST
                             <1>       ;MOV   DX,HF_REG_PORT
000045BC 668B15[EA5C0000]    <1>       MOV    DX,[HF_REG_PORT]
000045C3 EE                  <1>       OUT    DX,AL               ; RESET
                             <1> ;     MOV    CX,10               ; DELAY COUNT
                             <1> ;DRD: DEC    CX
                             <1> ;     JNZ    short DRD           ; WAIT 4.8 MICRO-SEC
                             <1>       ;mov   cx,2                ; wait for 30 micro seconds
000045C4 B902000000          <1>         mov ecx, 2 ; 21/02/2015
000045C9 E81FD8FFFF          <1>       call    WAITF                  ; (Award Bios 1999 - WAIT_REFRESH,
                             <1>                                        ; 40 micro seconds)
000045CE A0[D5580100]        <1>       mov    al,[CONTROL_BYTE]
000045D3 240F                <1>       AND    AL,0FH              ; SET HEAD OPTION
000045D5 EE                  <1>       OUT    DX,AL               ; TURN RESET OFF
000045D6 E838040000          <1>       CALL   NOT_BUSY
000045DB 7515                <1>       JNZ    short DRERR         ; TIME OUT ON RESET
000045DD 668B15[E85C0000]    <1>       MOV    DX,[HF_PORT]
000045E4 FEC2                <1>       inc    dl  ; HF_PORT+1
                             <1>       ; 02/01/2015 - Award BIOS 1999 - AHDSK.ASM
                             <1>         ;mov     cl, 10
000045E6 B90A000000          <1>         mov     ecx, 10 ; 21/02/2015
                             <1> drst1:
000045EB EC                  <1>       IN     AL,DX               ; GET RESET STATUS
```

```
000045EC 3C01                 <1>       CMP    AL,1
                              <1>       ; 04/01/2015
000045EE 740A                 <1>       jz     short drst2
                              <1>       ;JNZ   short DRERR         ; BAD RESET STATUS
                              <1>              ; Drive/Head Register - bit 4
000045F0 E2F9                 <1>       loop   drst1
                              <1> DRERR:
000045F2 C605[D3580100]05     <1>       MOV    byte [DISK_STATUS1],BAD_RESET ; CARD FAILED
000045F9 C3                   <1>       RETn
                              <1> drst2:
                              <1>       ; 14/02/2015
000045FA 6689FA               <1>       mov    dx,di
                              <1> ;drst3:
                              <1> ;     ; 05/01/2015
                              <1> ;     shl    di,1
                              <1> ;     ; 04/01/2015
                              <1> ;     mov    ax,[di+hd_cports]
                              <1> ;     cmp    ax,[HF_REG_PORT]
                              <1> ;     je     short drst4
                              <1> ;     mov    [HF_REG_PORT], ax
                              <1> ;     ; 03/01/2015
                              <1> ;     mov    ax,[di+hd_ports]
                              <1> ;       mov      [HF_PORT], ax
                              <1> ;     ; 05/01/2014
                              <1> ;     shr    di,1
                              <1> ;     ; 04/01/2015
                              <1> ;     jmp    short drst0  ; reset other controller
                              <1> ;drst4:
                              <1> ;     ; 05/01/2015
                              <1> ;     shr    di,1
                              <1> ;     mov    al,[di+hd_dregs]
                              <1> ;     and    al,10h ; bit 4 only
                              <1> ;     shr    al,4 ; bit 4  -> bit 0
                              <1> ;     mov    [hf_m_s], al ; (0 = master, 1 = slave)
                              <1>       ;
000045FD A0[EC5C0000]         <1>       mov    al, [hf_m_s] ; 18/01/2015
00004602 A801                 <1>       test   al,1
                              <1> ;     jnz    short drst6
00004604 7516                 <1>         jnz     short drst4
00004606 8065FDEF             <1>       AND     byte [CMD_BLOCK+5],0EFH ; SET TO DRIVE 0
                              <1> ;drst5:
                              <1> drst3:
0000460A E835010000           <1>       CALL   INIT_DRV            ; SET MAX HEADS
                              <1>       ;mov   dx,di
0000460F E8ED010000           <1>       CALL   HDISK_RECAL         ; RECAL TO RESET SEEK SPEED
                              <1>       ; 04/01/2014
                              <1> ;     inc    di
                              <1> ;     mov    dx,di
                              <1> ;     cmp    dl,[HF_NUM]
                              <1> ;     jb     short drst3
                              <1> ;DRE:
00004614 C605[D3580100]00     <1>       MOV    byte [DISK_STATUS1],0     ; IGNORE ANY SET UP ERRORS
0000461B C3                   <1>       RETn
                              <1> ;drst6:
                              <1> drst4:                ; Drive/Head Register - bit 4
0000461C 804DFD10             <1>       OR      byte [CMD_BLOCK+5],010H ; SET TO DRIVE 1
                              <1>         ;jmp    short drst5
00004620 EBE8                 <1>         jmp     short drst3
                              <1>
                              <1> ;---------------------------------------
                              <1> ;     DISK STATUS ROUTINE  (AH = 01H) :
                              <1> ;---------------------------------------
                              <1>
                              <1> RETURN_STATUS:
00004622 A0[D3580100]         <1>       MOV    AL,[DISK_STATUS1]   ; OBTAIN PREVIOUS STATUS
00004627 C605[D3580100]00     <1>         MOV     byte [DISK_STATUS1],0   ; RESET STATUS
0000462E C3                   <1>       RETn
                              <1>
                              <1> ;---------------------------------------
                              <1> ;     DISK READ ROUTINE    (AH = 02H) :
                              <1> ;---------------------------------------
                              <1>
                              <1> DISK_READ:
0000462F C645FE20             <1>       MOV    byte [CMD_BLOCK+6],READ_CMD
00004633 E954020000           <1>         JMP     COMMANDI
                              <1>
                              <1> ;---------------------------------------
                              <1> ;     DISK WRITE ROUTINE   (AH = 03H) :
                              <1> ;---------------------------------------
                              <1>
                              <1> DISK_WRITE:
00004638 C645FE30             <1>       MOV    byte [CMD_BLOCK+6],WRITE_CMD
0000463C E9A6020000           <1>         JMP     COMMANDO
                              <1>
                              <1> ;---------------------------------------
                              <1> ;     DISK VERIFY       (AH = 04H) :
                              <1> ;---------------------------------------
                              <1>
                              <1> DISK_VERF:
00004641 C645FE40             <1>       MOV    byte [CMD_BLOCK+6],VERIFY_CMD
00004645 E814030000           <1>       CALL   COMMAND
0000464A 750C                 <1>       JNZ    short VERF_EXIT           ; CONTROLLER STILL BUSY
0000464C E886030000           <1>       CALL   _WAIT               ; (Original: CALL WAIT)
00004651 7505                 <1>       JNZ    short VERF_EXIT           ; TIME OUT
00004653 E813040000           <1>       CALL   CHECK_STATUS
                              <1> VERF_EXIT:
00004658 C3                   <1>       RETn
                              <1>
                              <1> ;---------------------------------------
                              <1> ;     FORMATTING        (AH = 05H) :
                              <1> ;---------------------------------------
                              <1>
                              <1> FMT_TRK:                        ; FORMAT TRACK      (AH = 005H)
```

```
00004659 C645FE50            <1>       MOV    byte [CMD_BLOCK+6],FMTTRK_CMD
                             <1>       ;PUSH  ES
                             <1>       ;PUSH  BX
0000465D 53                  <1>       push   ebx
0000465E E8BA040000          <1>       CALL   GET_VEC             ; GET DISK PARAMETERS ADDRESS
                             <1>       ;MOV   AL,[ES:BX+14]       ; GET SECTORS/TRACK
00004663 8A430E              <1>       mov    al, [ebx+14]
00004666 8845F9              <1>       MOV    [CMD_BLOCK+1],AL    ; SET SECTOR COUNT IN COMMAND
00004669 5B                  <1>       pop    ebx
                             <1>       ;POP   BX
                             <1>       ;POP   ES
0000466A E97F020000          <1>         JMP     CMD_OF                   ; GO EXECUTE THE COMMAND
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     READ DASD TYPE           (AH = 15H) :
                             <1> ;----------------------------------------
                             <1>
                             <1> READ_DASD_TYPE:
                             <1> READ_D_T:                          ; GET DRIVE PARAMETERS
0000466F 1E                  <1>       PUSH   DS                  ; SAVE REGISTERS
                             <1>       ;PUSH  ES
00004670 53                  <1>       PUSH   eBX
                             <1>       ;CALL  DDS                 ; ESTABLISH ADDRESSING
                             <1>       ;push  cs
                             <1>       ;pop   ds
00004671 66BB1000            <1>         mov  bx, KDATA
00004675 8EDB                <1>       mov    ds, bx
                             <1>       ;mov   es, bx
00004677 C605[D3580100]00    <1>       MOV     byte [DISK_STATUS1],0
0000467E 8A1D[D4580100]      <1>       MOV    BL,[HF_NUM]         ; GET NUMBER OF DRIVES
00004684 80E27F              <1>       AND    DL,7FH              ; GET DRIVE NUMBER
00004687 38D3                <1>       CMP    BL,DL
00004689 7627                <1>       JBE    short RDT_NOT_PRESENT      ; RETURN DRIVE NOT PRESENT
0000468B E88D040000          <1>       CALL   GET_VEC             ; GET DISK PARAMETER ADDRESS
                             <1>       ;MOV   AL,[ES:BX+2]        ; HEADS
00004690 8A4302              <1>       mov    al, [ebx+2]
                             <1>       ;MOV   CL,[ES:BX+14]
00004693 8A4B0E              <1>       mov    cl, [ebx+14]
00004696 F6E9                <1>       IMUL   CL                  ; * NUMBER OF SECTORS
                             <1>       ;MOV   CX,[ES:BX]          ; MAX NUMBER OF CYLINDERS
00004698 668B0B              <1>       mov    cx ,[ebx]
                             <1>       ;
                             <1>       ; 02/01/2015
                             <1>       ; ** leave the last cylinder as reserved for diagnostics **
                             <1>       ; (Also in Award BIOS - 1999, AHDSK.ASM, FUN15 -> sub ax, 1)
0000469B 6649                <1>       DEC    CX                  ; LEAVE ONE FOR DIAGNOSTICS
                             <1>       ;
0000469D 66F7E9              <1>       IMUL   CX                  ; NUMBER OF SECTORS
000046A0 6689D1              <1>       MOV    CX,DX               ; HIGH ORDER HALF
000046A3 6689C2              <1>       MOV    DX,AX               ; LOW ORDER HALF
                             <1>       ;SUB   AX,AX
000046A6 28C0                <1>       sub    al, al
000046A8 B403                <1>       MOV    AH,03H              ; INDICATE FIXED DISK
000046AA 5B                  <1> RDT2: POP    eBX                 ; RESTORE REGISTERS
                             <1>       ;POP   ES
000046AB 1F                  <1>       POP    DS
                             <1>       ; (*) CLC                  ; CLEAR CARRY
                             <1>       ;RETf  2
                             <1>       ; (*) 29/05/2016
                             <1>       ; (*) retf 4
000046AC 80642408FE          <1>       and    byte [esp+8], 0FEh ; clear carry bit of eflags register
000046B1 CF                  <1>       iretd
                             <1>
                             <1> RDT_NOT_PRESENT:
000046B2 6629C0              <1>       SUB    AX,AX               ; DRIVE NOT PRESENT RETURN
000046B5 6689C1              <1>       MOV    CX,AX               ; ZERO BLOCK COUNT
000046B8 6689C2              <1>       MOV    DX,AX
000046BB EBED                <1>       JMP    short RDT2
                             <1>
                             <1> ; 28/05/2016
                             <1> ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     GET PARAMETERS           (AH = 08H) :
                             <1> ;----------------------------------------
                             <1>
                             <1> GET_PARM_N:
                             <1>       ; ebx = user's buffer address for parameters table
                             <1> ;GET_PARM:                         ; GET DRIVE PARAMETERS
000046BD 1E                  <1>       PUSH   DS                  ; SAVE REGISTERS
000046BE 06                  <1>       PUSH   ES
000046BF 53                  <1>       PUSH   eBX
                             <1>       ;MOV   AX,ABS0             ; ESTABLISH ADDRESSING
                             <1>       ;MOV   DS,AX
                             <1>       ;TEST  DL,1                ; CHECK FOR DRIVE 1
                             <1>       ;JZ    short G0
                             <1>       ;LES   BX,@HF1_TBL_VEC
                             <1>       ;JMP   SHORT G1
                             <1> ;G0:  LES    BX,@HF_TBL_VEC
                             <1> ;G1:
                             <1>       ;CALL  DDS                 ; ESTABLISH SEGMENT
                             <1>       ; 22/12/2014
                             <1>       ;push  cs
                             <1>       ;pop   ds
000046C0 66BB1000            <1>       mov    bx, KDATA
000046C4 8EDB                <1>       mov    ds, bx
000046C6 8EC3                <1>       mov    es, bx ; 27/05/2016
                             <1>       ;
000046C8 80EA80              <1>       SUB    DL,80H
000046CB 80FA04              <1>       CMP    DL,MAX_FILE         ; TEST WITHIN RANGE
000046CE 7361                <1>       JAE    short G4
                             <1>       ;
000046D0 31DB                <1>       xor    ebx, ebx ; 21/02/2015
```

```
                               <1>        ; 22/12/2014
000046D2 88D3                  <1>        mov    bl, dl
                               <1>        ;xor   bh, bh
000046D4 C0E302                <1>        shl    bl, 2                ; convert index to offset
                               <1>        ;add   bx, HF_TBL_VEC
000046D7 81C3[D8580100]        <1>        add    ebx, HF_TBL_VEC
                               <1>        ;mov   ax, [bx+2]
                               <1>        ;mov   es, ax               ; dpt segment
                               <1>        ;mov   bx, [bx]             ; dpt offset
000046DD 8B1B                  <1>        mov    ebx, [ebx] ; 32 bit offset
                               <1>
000046DF C605[D3580100]00      <1>        MOV    byte [DISK_STATUS1],0
                               <1>          ;MOV     AX,[ES:BX]              ; MAX NUMBER OF CYLINDERS
000046E6 668B03                <1>        mov    ax, [ebx]
                               <1>        ;;SUB  AX,2                 ; ADJUST FOR 0-N
000046E9 6648                  <1>        dec    ax                   ; max. cylinder number
000046EB 88C5                  <1>        MOV    CH,AL
000046ED 66250003              <1>        AND    AX,0300H             ; HIGH TWO BITS OF CYLINDER
000046F1 66D1E8                <1>        SHR    AX,1
000046F4 66D1E8                <1>        SHR    AX,1
                               <1>        ;OR    AL,[ES:BX+14]        ; SECTORS
000046F7 0A430E                <1>        or     al, [ebx+14]
000046FA 88C1                  <1>        MOV    CL,AL
                               <1>        ;MOV   DH,[ES:BX+2]         ; HEADS
000046FC 8A7302                <1>        mov    dh, [ebx+2]
000046FF FECE                  <1>        DEC    DH                   ; 0-N RANGE
00004701 8A15[D4580100]        <1>        MOV    DL,[HF_NUM]          ; DRIVE COUNT
00004707 6629C0                <1>        SUB    AX,AX
                               <1>          ;27/12/2014
                               <1>        ;mov   di, bx               ; HDPT offset
                               <1>
                               <1>        ; 27/05/2016
                               <1>        ; return fixed disk parameters table to user
                               <1>        ; in user's buffer, which is pointed by EBX
                               <1>        ;
0000470A 873C24                <1>        xchg   edi, [esp]           ; ebx (input)-> edi, edi -> [esp]
0000470D 56                    <1>        push   esi
0000470E 89DE                  <1>        mov    esi, ebx             ; hard disk parameter table (32 bytes)
00004710 89FB                  <1>        mov    ebx, edi             ; ebx = user's buffer address
00004712 51                    <1>        push   ecx
00004713 50                    <1>        push   eax
00004714 B920000000            <1>        mov    ecx, 32 ; 32 bytes
00004719 E85BA00000            <1>        call   transfer_to_user_buffer ; trdosk6.s (16/05/2016)
0000471E 58                    <1>        pop    eax
0000471F 59                    <1>        pop    ecx
00004720 5E                    <1>        pop    esi
00004721 5F                    <1>        pop    edi
00004722 730A                  <1>        jnc    short G5
                               <1>        ; 29/05/2016 (*)
00004724 B8FF000000            <1>        mov    eax, 0FFh ; unknown error !
                               <1> _G6:
00004729 804C241001            <1>        or     byte [esp+16], 1 ; set carry bit of eflags register
                               <1> G5:
                               <1>        ; 27/05/2016
                               <1>        ;POP   eBX                  ; RESTORE REGISTERS
0000472E 07                    <1>        POP    ES
0000472F 1F                    <1>        POP    DS
                               <1>        ;RETf  2
                               <1>        ; (*) 29/05/2016
                               <1>        ; (*) retf 4
                               <1>        ; (*) or byte [esp+8], 1 ; set carry bit of eflags register
00004730 CF                    <1>        iretd
                               <1> G4:
00004731 C605[D3580100]07      <1>        MOV     byte [DISK_STATUS1],INIT_FAIL ; OPERATION FAILED
00004738 B407                  <1>        MOV    AH,INIT_FAIL
0000473A 28C0                  <1>        SUB    AL,AL
0000473C 6629D2                <1>        SUB    DX,DX
0000473F 6629C9                <1>        SUB    CX,CX
                               <1>        ; 29/05/2016 (*)
                               <1>        ;STC                        ; SET ERROR FLAG
                               <1>        ;JMP   short G5
00004742 EBE5                  <1>        jmp    short _G6
                               <1>
                               <1> ;------------------------------------------
                               <1> ;     INITIALIZE DRIVE      (AH = 09H) :
                               <1> ;------------------------------------------
                               <1>        ; 03/01/2015
                               <1>        ; According to ATA-ATAPI specification v2.0 to v5.0
                               <1>        ; logical sector per logical track
                               <1>        ; and logical heads - 1 would be set but
                               <1>        ; it is seen as it will be good
                               <1>        ; if physical parameters will be set here
                               <1>        ; because, number of heads <= 16.
                               <1>        ; (logical heads usually more than 16)
                               <1>        ; NOTE: ATA logical parameters (software C, H, S)
                               <1>        ;      == INT 13h physical parameters
                               <1>
                               <1> ;INIT_DRV:
                               <1> ;      MOV    byte [CMD_BLOCK+6],SET_PARM_CMD
                               <1> ;      CALL   GET_VEC              ; ES:BX -> PARAMETER BLOCK
                               <1> ;      MOV    AL,[ES:BX+2]         ; GET NUMBER OF HEADS
                               <1> ;      DEC    AL                   ; CONVERT TO 0-INDEX
                               <1> ;      MOV    AH,[CMD_BLOCK+5]     ; GET SDH REGISTER
                               <1> ;      AND    AH,0F0H              ; CHANGE HEAD NUMBER
                               <1> ;      OR     AH,AL                ; TO MAX HEAD
                               <1> ;      MOV    [CMD_BLOCK+5],AH
                               <1> ;      MOV    AL,[ES:BX+14]        ; MAX SECTOR NUMBER
                               <1> ;      MOV    [CMD_BLOCK+1],AL
                               <1> ;      SUB    AX,AX
                               <1> ;      MOV    [CMD_BLOCK+3],AL     ; ZERO FLAGS
                               <1> ;      CALL   COMMAND              ; TELL CONTROLLER
                               <1> ;      JNZ    short INIT_EXIT               ; CONTROLLER BUSY ERROR
                               <1> ;      CALL   NOT_BUSY             ; WAIT FOR IT TO BE DONE
```

```
                             <1> ;     JNZ    short INIT_EXIT          ; TIME OUT
                             <1> ;     CALL   CHECK_STATUS
                             <1> ;INIT_EXIT:
                             <1> ;     RETn
                             <1>
                             <1> ; 04/01/2015
                             <1> ; 02/01/2015 - Derived from from AWARD BIOS 1999
                             <1> ;                          AHDSK.ASM - INIT_DRIVE
                             <1> INIT_DRV:
                             <1>       ;xor   ah,ah
00004744 31C0                <1>       xor    eax, eax ; 21/02/2015
00004746 B00B                <1>       mov    al,11 ; Physical heads from translated HDPT
00004748 3825[E8580100]      <1>         cmp     [LBAMode], ah   ; 0
0000474E 7702                <1>       ja     short idrv0
00004750 B002                <1>       mov    al,2  ; Physical heads from standard HDPT
                             <1> idrv0:
                             <1>       ; DL = drive number (0 based)
00004752 E8C6030000          <1>       call   GET_VEC
                             <1>       ;push  bx
00004757 53                  <1>       push   ebx ; 21/02/2015
                             <1>       ;add   bx,ax
00004758 01C3                <1>       add    ebx, eax
                             <1>       ;; 05/01/2015
0000475A 8A25[EC5C0000]      <1>       mov    ah, [hf_m_s] ; drive number (0= master, 1= slave)
                             <1>       ;;and  ah,1
00004760 C0E404              <1>       shl    ah,4
00004763 80CCA0              <1>       or     ah,0A0h  ; Drive/Head register - 10100000b (A0h)
                             <1>       ;mov   al,[es:bx]
00004766 8A03                <1>       mov    al, [ebx] ; 21/02/2015
00004768 FEC8                <1>       dec    al      ; last head number
                             <1>       ;and   al,0Fh
0000476A 08E0                <1>       or     al,ah   ; lower 4 bits for head number
                             <1>       ;
0000476C C645FE91            <1>       mov    byte [CMD_BLOCK+6],SET_PARM_CMD
00004770 8845FD              <1>       mov    [CMD_BLOCK+5],al
                             <1>       ;pop   bx
00004773 5B                  <1>       pop    ebx
00004774 29C0                <1>       sub    eax, eax ; 21/02/2015
00004776 B004                <1>       mov    al,4 ; Physical sec per track from translated HDPT
00004778 803D[E8580100]00    <1>       cmp    byte [LBAMode], 0
0000477F 7702                <1>       ja     short idrv1
00004781 B00E                <1>       mov    al,14 ; Physical sec per track from standard HDPT
                             <1> idrv1:
                             <1>       ;xor   ah,ah
                             <1>       ;add   bx,ax
00004783 01C3                <1>       add    ebx, eax ; 21/02/2015
                             <1>       ;mov   al,[es:bx]
                             <1>                   ; sector number
00004785 8A03                <1>       mov    al, [ebx]
00004787 8845F9              <1>       mov    [CMD_BLOCK+1],al
0000478A 28C0                <1>       sub    al,al
0000478C 8845FB              <1>       mov    [CMD_BLOCK+3],al  ; ZERO FLAGS
0000478F E8CA010000          <1>       call   COMMAND        ; TELL CONTROLLER
00004794 750C                <1>       jnz    short INIT_EXIT       ; CONTROLLER BUSY ERROR
00004796 E878020000          <1>       call   NOT_BUSY       ; WAIT FOR IT TO BE DONE
0000479B 7505                <1>       jnz    short INIT_EXIT       ; TIME OUT
0000479D E8C9020000          <1>       call   CHECK_STATUS
                             <1> INIT_EXIT:
000047A2 C3                  <1>       RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     READ LONG         (AH = 0AH) :
                             <1> ;----------------------------------------
                             <1>
                             <1> RD_LONG:
                             <1>       ;MOV   @CMD_BLOCK+6,READ_CMD OR ECC_MODE
000047A3 C645FE22            <1>         mov     byte [CMD_BLOCK+6],READ_CMD + ECC_MODE
000047A7 E9E0000000          <1>         JMP     COMMANDI
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     WRITE LONG        (AH = 0BH) :
                             <1> ;----------------------------------------
                             <1>
                             <1> WR_LONG:
                             <1>       ;MOV   @CMD_BLOCK+6,WRITE_CMD OR ECC_MODE
000047AC C645FE32            <1>         MOV     byte [CMD_BLOCK+6],WRITE_CMD + ECC_MODE
000047B0 E932010000          <1>         JMP     COMMANDO
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     SEEK              (AH = 0CH) :
                             <1> ;----------------------------------------
                             <1>
                             <1> DISK_SEEK:
000047B5 C645FE70            <1>         MOV     byte [CMD_BLOCK+6],SEEK_CMD
000047B9 E8A0010000          <1>       CALL   COMMAND
000047BE 751C                <1>       JNZ    short DS_EXIT             ; CONTROLLER BUSY ERROR
000047C0 E812020000          <1>       CALL   _WAIT
000047C5 7515                <1>         JNZ     DS_EXIT                 ; TIME OUT ON SEEK
000047C7 E89F020000          <1>       CALL   CHECK_STATUS
000047CC 803D[D3580100]40    <1>         CMP     byte [DISK_STATUS1],BAD_SEEK
000047D3 7507                <1>       JNE    short DS_EXIT
000047D5 C605[D3580100]00    <1>         MOV     byte [DISK_STATUS1],0
                             <1> DS_EXIT:
000047DC C3                  <1>       RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     TEST DISK READY      (AH = 10H) :
                             <1> ;----------------------------------------
                             <1>
                             <1> TST_RDY:                        ; WAIT FOR CONTROLLER
000047DD E831020000          <1>       CALL   NOT_BUSY
000047E2 751C                <1>       JNZ    short TR_EX
000047E4 8A45FD              <1>       MOV    AL,[CMD_BLOCK+5]    ; SELECT DRIVE
```

```
000047E7 668B15[E85C0000]    <1>       MOV    DX,[HF_PORT]
000047EE 80C206              <1>       add    dl,6
000047F1 EE                  <1>       OUT    DX,AL
000047F2 E88C020000          <1>       CALL   CHECK_ST           ; CHECK STATUS ONLY
000047F7 7507                <1>       JNZ    short TR_EX
000047F9 C605[D3580100]00    <1>       MOV    byte [DISK_STATUS1],0     ; WIPE OUT DATA CORRECTED ERROR
                             <1> TR_EX:
00004800 C3                  <1>       RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     RECALIBRATE       (AH = 11H) :
                             <1> ;----------------------------------------
                             <1>
                             <1> HDISK_RECAL:
00004801 C645FE10            <1>         MOV     byte [CMD_BLOCK+6],RECAL_CMD ; 10h, 16
00004805 E854010000          <1>       CALL   COMMAND            ; START THE OPERATION
0000480A 7523                <1>       JNZ    short RECAL_EXIT   ; ERROR
0000480C E8C6010000          <1>       CALL   _WAIT              ; WAIT FOR COMPLETION
00004811 7407                <1>       JZ     short RECAL_X             ; TIME OUT ONE OK ?
00004813 E8BF010000          <1>       CALL   _WAIT              ; WAIT FOR COMPLETION LONGER
00004818 7515                <1>       JNZ    short RECAL_EXIT   ; TIME OUT TWO TIMES IS ERROR
                             <1> RECAL_X:
0000481A E84C020000          <1>       CALL   CHECK_STATUS
0000481F 803D[D3580100]40    <1>       CMP    byte [DISK_STATUS1],BAD_SEEK ; SEEK NOT COMPLETE
00004826 7507                <1>       JNE    short RECAL_EXIT   ; IS OK
00004828 C605[D3580100]00    <1>       MOV    byte [DISK_STATUS1],0
                             <1> RECAL_EXIT:
0000482F 803D[D3580100]00    <1>         CMP     byte [DISK_STATUS1],0
00004836 C3                  <1>       RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ;      CONTROLLER DIAGNOSTIC (AH = 14H) :
                             <1> ;----------------------------------------
                             <1>
                             <1> CTLR_DIAGNOSTIC:
00004837 FA                  <1>         CLI                             ; DISABLE INTERRUPTS WHILE CHANGING MASK
00004838 E4A1                <1>       IN     AL,INTB01          ; TURN ON SECOND INTERRUPT CHIP
                             <1>       ;AND   AL,0BFH
0000483A 243F                <1>       and    al, 3Fh                   ; enable IRQ 14 & IRQ 15
                             <1>       ;JMP   $+2
                             <1>       IODELAY
0000483C EB00                <2>  jmp short $+2
0000483E EB00                <2>  jmp short $+2
00004840 E6A1                <1>       OUT    INTB01,AL
                             <1>       IODELAY
00004842 EB00                <2>  jmp short $+2
00004844 EB00                <2>  jmp short $+2
00004846 E421                <1>       IN     AL,INTA01          ; LET INTERRUPTS PASS THRU TO
00004848 24FB                <1>       AND    AL,0FBH            ;  SECOND CHIP
                             <1>       ;JMP   $+2
                             <1>       IODELAY
0000484A EB00                <2>  jmp short $+2
0000484C EB00                <2>  jmp short $+2
0000484E E621                <1>       OUT    INTA01,AL
00004850 FB                  <1>       STI
00004851 E8BD010000          <1>       CALL   NOT_BUSY           ; WAIT FOR CARD
00004856 752B                <1>       JNZ    short CD_ERR       ; BAD CARD
                             <1>       ;MOV   DX, HF_PORT+7
00004858 668B15[E85C0000]    <1>       mov    dx, [HF_PORT]
0000485F 80C207              <1>       add    dl, 7
00004862 B090                <1>       MOV    AL,DIAG_CMD        ; START DIAGNOSE
00004864 EE                  <1>       OUT    DX,AL
00004865 E8A9010000          <1>       CALL   NOT_BUSY           ; WAIT FOR IT TO COMPLETE
0000486A B480                <1>       MOV    AH,TIME_OUT
0000486C 7517                <1>       JNZ    short CD_EXIT             ; TIME OUT ON DIAGNOSTIC
                             <1>       ;MOV   DX,HF_PORT+1       ; GET ERROR REGISTER
0000486E 668B15[E85C0000]    <1>       mov    dx, [HF_PORT]
00004875 FEC2                <1>       inc    dl
00004877 EC                  <1>       IN     AL,DX
00004878 A2[CA580100]        <1>       MOV    [HF_ERROR],AL      ; SAVE IT
0000487D B400                <1>       MOV    AH,0
0000487F 3C01                <1>       CMP    AL,1               ; CHECK FOR ALL OK
00004881 7402                <1>       JE     SHORT CD_EXIT
00004883 B420                <1> CD_ERR: MOV  AH,BAD_CNTLR
                             <1> CD_EXIT:
00004885 8825[D3580100]      <1>       MOV    [DISK_STATUS1],AH
0000488B C3                  <1>       RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ; COMMANDI                          :
                             <1> ;     REPEATEDLY INPUTS DATA TILL      :
                             <1> ;     NSECTOR RETURNS ZERO             :
                             <1> ;----------------------------------------
                             <1> COMMANDI:
0000488C E862020000          <1>       CALL   CHECK_DMA          ; CHECK 64K BOUNDARY ERROR
00004891 7253                <1>       JC     short CMD_ABORT
                             <1>       ;MOV   DI,BX
00004893 89DF                <1>       mov    edi, ebx ; 21/02/2015
00004895 E8C4000000          <1>       CALL   COMMAND            ; OUTPUT COMMAND
0000489A 754A                <1>       JNZ    short CMD_ABORT
                             <1> CMD_I1:
0000489C E836010000          <1>       CALL   _WAIT              ; WAIT FOR DATA REQUEST INTERRUPT
000048A1 7543                <1>       JNZ    short TM_OUT       ; TIME OUT
                             <1> cmd_i1x: ; 18/02/2016
                             <1>       ;MOV   CX,256             ; SECTOR SIZE IN WORDS
000048A3 B900010000          <1>       mov    ecx, 256 ; 21/02/2015
                             <1>       ;MOV   DX,HF_PORT
000048A8 668B15[E85C0000]    <1>       mov    dx,[HF_PORT]
000048AF FA                  <1>       CLI
000048B0 FC                  <1>       CLD
000048B1 F3666D              <1>       REP    INSW               ; GET THE SECTOR
000048B4 FB                  <1>       STI
000048B5 F645FE02            <1>       TEST   byte [CMD_BLOCK+6],ECC_MODE ; CHECK FOR NORMAL INPUT
```

```
000048B9 7419                <1>       JZ     short CMD_I3
000048BB E880010000          <1>       CALL   WAIT_DRQ            ; WAIT FOR DATA REQUEST
000048C0 7224                <1>       JC     short TM_OUT
                             <1>       ;MOV   DX,HF_PORT
000048C2 668B15[E85C0000]    <1>       mov    dx,[HF_PORT]
                             <1>       ;MOV   CX,4                ; GET ECC BYTES
000048C9 B904000000          <1>       mov    ecx, 4 ; mov cx, 4
000048CE EC                  <1> CMD_I2: IN   AL,DX
                             <1>       ;MOV   [ES:DI],AL          ; GO SLOW FOR BOARD
000048CF 8807                <1>       mov    [edi], al ; 21/02/2015
000048D1 47                  <1>       INC    eDI
000048D2 E2FA                <1>       LOOP   CMD_I2
                             <1> CMD_I3:
                             <1>       ; wait for 400 ns
000048D4 80C207              <1>       add    dl, 7
000048D7 EC                  <1>       in     al, dx
000048D8 EC                  <1>       in     al, dx
000048D9 EC                  <1>       in     al, dx
                             <1>       ;
000048DA E88C010000          <1>       CALL   CHECK_STATUS
000048DF 7505                <1>       JNZ    short CMD_ABORT           ; ERROR RETURNED
000048E1 FE4DF9              <1>       DEC    byte [CMD_BLOCK+1]  ; CHECK FOR MORE
                             <1>       ;JNZ   SHORT CMD_I1
000048E4 75BD                <1>       jnz    short cmd_i1x ; 18/02/2016
                             <1> CMD_ABORT:
000048E6 C3                  <1> TM_OUT: RETn
                             <1>
                             <1> ;---------------------------------------
                             <1> ; COMMANDO                            :
                             <1> ;     REPEATEDLY OUTPUTS DATA TILL     :
                             <1> ;     NSECTOR RETURNS ZERO             :
                             <1> ;---------------------------------------
                             <1> COMMANDO:
000048E7 E807020000          <1>       CALL   CHECK_DMA           ; CHECK 64K BOUNDARY ERROR
000048EC 72F8                <1>       JC     short CMD_ABORT
000048EE 89DE                <1> CMD_OF: MOV eSI,eBX ; 21/02/2015
000048F0 E869000000          <1>       CALL   COMMAND             ; OUTPUT COMMAND
000048F5 75EF                <1>       JNZ    short CMD_ABORT
000048F7 E844010000          <1>       CALL   WAIT_DRQ            ; WAIT FOR DATA REQUEST
000048FC 72E8                <1>       JC     short TM_OUT              ; TOO LONG
                             <1> CMD_O1: ;PUSH      DS
                             <1>       ;PUSH  ES                  ; MOVE ES TO DS
                             <1>       ;POP   DS
                             <1>       ;MOV   CX,256              ; PUT THE DATA OUT TO THE CARD
                             <1>       ;MOV   DX,HF_PORT
                             <1>       ; 01/02/2015
000048FE 668B15[E85C0000]    <1>       mov    dx, [HF_PORT]
                             <1>       ;push  es
                             <1>       ;pop   ds
                             <1>       ;mov   cx, 256
00004905 B900010000          <1>       mov    ecx, 256 ; 21/02/2015
0000490A FA                  <1>       CLI
0000490B FC                  <1>       CLD
0000490C F3666F              <1>       REP    OUTSW
0000490F FB                  <1>       STI
                             <1>       ;POP   DS                  ; RESTORE DS
00004910 F645FE02            <1>       TEST   byte [CMD_BLOCK+6],ECC_MODE ; CHECK FOR NORMAL OUTPUT
00004914 7419                <1>       JZ     short CMD_O3
00004916 E825010000          <1>       CALL   WAIT_DRQ            ; WAIT FOR DATA REQUEST
0000491B 72C9                <1>       JC     short TM_OUT
                             <1>       ;MOV   DX,HF_PORT
0000491D 668B15[E85C0000]    <1>       mov    dx, [HF_PORT]
                             <1>       ;MOV   CX,4                ; OUTPUT THE ECC BYTES
00004924 B904000000          <1>       mov    ecx, 4  ; mov cx, 4
                             <1> CMD_O2: ;MOV AL,[ES:SI]
00004929 8A06                <1>       mov    al, [esi]
0000492B EE                  <1>       OUT    DX,AL
0000492C 46                  <1>       INC    eSI
0000492D E2FA                <1>       LOOP   CMD_O2
                             <1> CMD_O3:
0000492F E8A3000000          <1>       CALL   _WAIT               ; WAIT FOR SECTOR COMPLETE INTERRUPT
00004934 75B0                <1>       JNZ    short TM_OUT        ; ERROR RETURNED
00004936 E830010000          <1>       CALL   CHECK_STATUS
0000493B 75A9                <1>       JNZ    short CMD_ABORT
0000493D F605[C9580100]08    <1>       TEST   byte [HF_STATUS],ST_DRQ    ; CHECK FOR MORE
00004944 75B8                <1>       JNZ    SHORT CMD_O1
                             <1>       ;MOV   DX,HF_PORT+2        ; CHECK RESIDUAL SECTOR COUNT
00004946 668B15[E85C0000]    <1>       mov    dx, [HF_PORT]
                             <1>       ;add   dl, 2
0000494D FEC2                <1>       inc    dl
0000494F FEC2                <1>       inc    dl
00004951 EC                  <1>       IN     AL,DX               ;
00004952 A8FF                <1>       TEST   AL,0FFH             ;
00004954 7407                <1>       JZ     short CMD_O4              ; COUNT = 0  OK
00004956 C605[D3580100]BB    <1>       MOV    byte [DISK_STATUS1],UNDEF_ERR
                             <1>                                  ; OPERATION ABORTED - PARTIAL TRANSFER
                             <1> CMD_O4:
0000495D C3                  <1>       RETn
                             <1>
                             <1> ;-------------------------------------------------------
                             <1> ; COMMAND                                      :
                             <1> ;     THIS ROUTINE OUTPUTS THE COMMAND BLOCK         :
                             <1> ; OUTPUT                                       :
                             <1> ;     BL = STATUS                              :
                             <1> ;     BH = ERROR REGISTER                      :
                             <1> ;-------------------------------------------------------
                             <1>
                             <1> COMMAND:
0000495E 53                  <1>       PUSH   eBX                 ; WAIT FOR SEEK COMPLETE AND READY
                             <1>       ;;MOV  CX,DELAY_2          ; SET INITIAL DELAY BEFORE TEST
                             <1> COMMAND1:
                             <1>       ;;PUSH CX                  ; SAVE LOOP COUNT
0000495F E879FEFFFF          <1>       CALL   TST_RDY             ; CHECK DRIVE READY
```

```
                             <1>       ;;POP CX
00004964 7419                <1>       JZ    short COMMAND2             ; DRIVE IS READY
00004966 803D[D3580100]80    <1>         CMP     byte [DISK_STATUS1],TIME_OUT ; TST_RDY TIMED OUT--GIVE UP
                             <1>       ;JZ   short CMD_TIMEOUT
                             <1>       ;;LOOP COMMAND1           ; KEEP TRYING FOR A WHILE
                             <1>       ;JMP  SHORT COMMAND4             ; ITS NOT GOING TO GET READY
0000496D 7507                <1>       jne   short COMMAND4
                             <1> CMD_TIMEOUT:
0000496F C605[D3580100]20    <1>       MOV   byte [DISK_STATUS1],BAD_CNTLR
                             <1> COMMAND4:
00004976 5B                  <1>       POP   eBX
00004977 803D[D3580100]00    <1>         CMP     byte [DISK_STATUS1],0   ; SET CONDITION CODE FOR CALLER
0000497E C3                  <1>       RETn
                             <1> COMMAND2:
0000497F 5B                  <1>       POP   eBX
00004980 57                  <1>       PUSH  eDI
00004981 C605[CB580100]00    <1>       MOV   byte [HF_INT_FLAG],0       ; RESET INTERRUPT FLAG
00004988 FA                  <1>       CLI                       ; INHIBIT INTERRUPTS WHILE CHANGING MASK
00004989 E4A1                <1>       IN    AL,INTB01           ; TURN ON SECOND INTERRUPT CHIP
                             <1>       ;AND  AL,0BFH
0000498B 243F                <1>       and   al, 3Fh                    ; Enable IRQ 14 & 15
                             <1>       ;JMP  $+2
                             <1>       IODELAY
0000498D EB00                <2>  jmp short $+2
0000498F EB00                <2>  jmp short $+2
00004991 E6A1                <1>       OUT   INTB01,AL
00004993 E421                <1>       IN    AL,INTA01           ; LET INTERRUPTS PASS THRU TO
00004995 24FB                <1>       AND   AL,0FBH             ;  SECOND CHIP
                             <1>       ;JMP  $+2
                             <1>       IODELAY
00004997 EB00                <2>  jmp short $+2
00004999 EB00                <2>  jmp short $+2
0000499B E621                <1>       OUT   INTA01,AL
0000499D FB                  <1>       STI
0000499E 31FF                <1>       XOR   eDI,eDI                    ; INDEX THE COMMAND TABLE
                             <1>       ;MOV  DX,HF_PORT+1        ; DISK ADDRESS
000049A0 668B15[E85C0000]    <1>       mov   dx, [HF_PORT]
000049A7 FEC2                <1>       inc   dl
000049A9 F605[D5580100]C0    <1>       TEST  byte [CONTROL_BYTE],0C0H ; CHECK FOR RETRY SUPPRESSION
000049B0 7411                <1>       JZ    short COMMAND3
000049B2 8A45FE              <1>       MOV   AL, [CMD_BLOCK+6]   ; YES-GET OPERATION CODE
000049B5 24F0                <1>       AND   AL,0F0H             ; GET RID OF MODIFIERS
000049B7 3C20                <1>       CMP   AL,20H              ; 20H-40H IS READ, WRITE, VERIFY
000049B9 7208                <1>       JB    short COMMAND3
000049BB 3C40                <1>       CMP   AL,40H
000049BD 7704                <1>       JA    short COMMAND3
000049BF 804DFE01            <1>       OR    byte [CMD_BLOCK+6],NO_RETRIES
                             <1>                                 ; VALID OPERATION FOR RETRY SUPPRESS
                             <1> COMMAND3:
000049C3 8A443DF8            <1>       MOV   AL,[CMD_BLOCK+eDI]  ; GET THE COMMAND STRING BYTE
000049C7 EE                  <1>       OUT   DX,AL               ; GIVE IT TO CONTROLLER
                             <1>       IODELAY
000049C8 EB00                <2>  jmp short $+2
000049CA EB00                <2>  jmp short $+2
000049CC 47                  <1>       INC   eDI                 ; NEXT BYTE IN COMMAND BLOCK
000049CD 6642                <1>       INC   DX                  ; NEXT DISK ADAPTER REGISTER
000049CF 6683FF07            <1>       cmp   di, 7  ; 1/1/2015   ; ALL DONE?
000049D3 75EE                <1>       JNZ   short COMMAND3             ; NO--GO DO NEXT ONE
000049D5 5F                  <1>       POP   eDI
000049D6 C3                  <1>       RETn                      ; ZERO FLAG IS SET
                             <1> 
                             <1> ;CMD_TIMEOUT:
                             <1> ;     MOV   byte [DISK_STATUS1],BAD_CNTLR
                             <1> ;COMMAND4:
                             <1> ;     POP   BX
                             <1> ;     CMP   [DISK_STATUS1],0    ; SET CONDITION CODE FOR CALLER
                             <1> ;     RETn
                             <1> 
                             <1> ;----------------------------------------
                             <1> ;     WAIT FOR INTERRUPT        :
                             <1> ;----------------------------------------
                             <1> ;WAIT:
                             <1> _WAIT:
000049D7 FB                  <1>       STI                       ; MAKE SURE INTERRUPTS ARE ON
                             <1>       ;SUB  CX,CX               ; SET INITIAL DELAY BEFORE TEST
                             <1>       ;CLC
                             <1>       ;MOV  AX,9000H            ; DEVICE WAIT INTERRUPT
                             <1>       ;INT  15H
                             <1>       ;JC   WT2                 ; DEVICE TIMED OUT
                             <1>       ;MOV  BL,DELAY_1          ; SET DELAY COUNT
                             <1> 
                             <1>       ;mov  bl, WAIT_HDU_INT_HI
                             <1>       ;; 21/02/2015
                             <1>       ;;mov bl, WAIT_HDU_INT_HI + 1
                             <1>       ;;mov cx, WAIT_HDU_INT_LO
000049D8 B915160500          <1>       mov   ecx, WAIT_HDU_INT_LH
                             <1>                                 ; (AWARD BIOS -> WAIT_FOR_MEM)
                             <1> ;-----      WAIT LOOP
                             <1> 
                             <1> WT1:
                             <1>       ;TEST byte [HF_INT_FLAG],80H     ; TEST FOR INTERRUPT
000049DD F605[CB580100]C0    <1>       test  byte [HF_INT_FLAG],0C0h
                             <1>       ;LOOPZ WT1
000049E4 7517                <1>       JNZ   short WT3           ; INTERRUPT--LETS GO
                             <1>       ;DEC  BL
                             <1>       ;JNZ  short WT1           ; KEEP TRYING FOR A WHILE
                             <1> 
                             <1> WT1_hi:
000049E6 E461                <1>       in    al, SYS1 ; 61h (PORT_B)    ; wait for lo to hi
000049E8 A810                <1>       test  al, 10h                    ; transition on memory
000049EA 75FA                <1>       jnz   short WT1_hi        ; refresh.
                             <1> WT1_lo:
000049EC E461                <1>       in    al, SYS1            ; 061h (PORT_B)
```

```
000049EE A810                <1>       test  al, 10h
000049F0 74FA                <1>       jz    short WT1_lo
000049F2 E2E9                <1>       loop  WT1
                             <1>       ;;or  bl, bl
                             <1>       ;;jz  short WT2
                             <1>       ;;dec bl
                             <1>       ;;jmp short WT1
                             <1>       ;dec  bl
                             <1>       ;jnz  short WT1
                             <1>
000049F4 C605[D3580100]80    <1> WT2:  MOV   byte [DISK_STATUS1],TIME_OUT ; REPORT TIME OUT ERROR
000049FB EB0E                <1>       JMP   SHORT WT4
000049FD C605[D3580100]00    <1> WT3:  MOV   byte [DISK_STATUS1],0
00004A04 C605[CB580100]00    <1>       MOV   byte [HF_INT_FLAG],0
00004A0B 803D[D3580100]00    <1> WT4:  CMP   byte [DISK_STATUS1],0      ; SET CONDITION CODE FOR CALLER
00004A12 C3                  <1>       RETn
                             <1>
                             <1> ;---------------------------------------
                             <1> ;     WAIT FOR CONTROLLER NOT BUSY     :
                             <1> ;---------------------------------------
                             <1> NOT_BUSY:
00004A13 FB                  <1>       STI                       ; MAKE SURE INTERRUPTS ARE ON
                             <1>       ;PUSH eBX
                             <1>       ;SUB  CX,CX               ; SET INITIAL DELAY BEFORE TEST
00004A14 668B15[E85C0000]    <1>       mov   DX, [HF_PORT]
00004A1B 80C207              <1>       add   dl, 7               ; Status port (HF_PORT+7)
                             <1>       ;MOV  BL,DELAY_1
                             <1>                                 ; wait for 10 seconds
                             <1>       ;mov  cx, WAIT_HDU_INT_LO ; 1615h
                             <1>       ;;mov bl, WAIT_HDU_INT_HI ;   05h
                             <1>       ;mov  bl, WAIT_HDU_INT_HI + 1
00004A1E B915160500          <1>       mov   ecx, WAIT_HDU_INT_LH  ; 21/02/2015
                             <1>       ;
                             <1> ;;      mov     byte [wait_count], 0    ; Reset wait counter
                             <1> NB1:
00004A23 EC                  <1>       IN    AL,DX               ; CHECK STATUS
                             <1>       ;TEST AL,ST_BUSY
00004A24 2480                <1>       and   al, ST_BUSY
                             <1>       ;LOOPNZ      NB1
00004A26 7410                <1>       JZ    short NB2           ; NOT BUSY--LETS GO
                             <1>       ;DEC  BL
                             <1>       ;JNZ  short NB1           ; KEEP TRYING FOR A WHILE
                             <1>
00004A28 E461                <1> NB1_hi: IN  AL,SYS1                    ; wait for hi to lo
00004A2A A810                <1>       TEST  AL,010H                    ; transition on memory
00004A2C 75FA                <1>       JNZ   SHORT NB1_hi        ; refresh.
00004A2E E461                <1> NB1_lo: IN  AL,SYS1
00004A30 A810                <1>       TEST  AL,010H
00004A32 74FA                <1>       JZ    short NB1_lo
00004A34 E2ED                <1>       LOOP  NB1
                             <1>       ;dec  bl
                             <1>       ;jnz  short NB1
                             <1>       ;
                             <1> ;;      cmp     byte [wait_count], 182  ; 10 seconds (182 timer ticks)
                             <1> ;;    jb    short NB1
                             <1>       ;
                             <1>       ;MOV  [DISK_STATUS1],TIME_OUT    ; REPORT TIME OUT ERROR
                             <1>       ;JMP  SHORT NB3
00004A36 B080                <1>       mov   al, TIME_OUT
                             <1> NB2:
                             <1>       ;MOV  byte [DISK_STATUS1],0
                             <1> ;NB3:
                             <1>       ;POP  eBX
00004A38 A2[D3580100]        <1>       mov   [DISK_STATUS1], al  ;;; will be set after return
                             <1>       ;CMP  byte [DISK_STATUS1],0      ; SET CONDITION CODE FOR CALLER
00004A3D 08C0                <1>       or    al, al              ; (zf = 0 --> timeout)
00004A3F C3                  <1>       RETn
                             <1>
                             <1> ;---------------------------------------
                             <1> ;     WAIT FOR DATA REQUEST            :
                             <1> ;---------------------------------------
                             <1> WAIT_DRQ:
                             <1>       ;MOV  CX,DELAY_3
                             <1>       ;MOV  DX,HF_PORT+7
00004A40 668B15[E85C0000]    <1>       mov   dx, [HF_PORT]
00004A47 80C207              <1>       add   dl, 7
                             <1>       ;;MOV bl, WAIT_HDU_DRQ_HI ; 0
                             <1>       ;MOV  cx, WAIT_HDU_DRQ_LO ; 1000 (30 milli seconds)
                             <1>                                 ; (but it is written as 2000
                             <1>                                 ; micro seconds in ATORGS.ASM file
                             <1>                                 ; of Award Bios - 1999, D1A0622)
00004A4A B9E8030000          <1>       mov   ecx, WAIT_HDU_DRQ_LH ; 21/02/2015
00004A4F EC                  <1> WQ_1: IN    AL,DX               ; GET STATUS
00004A50 A808                <1>       TEST  AL,ST_DRQ           ; WAIT FOR DRQ
00004A52 7516                <1>       JNZ   short WQ_OK
                             <1>       ;LOOP WQ_1                ; KEEP TRYING FOR A SHORT WHILE
                             <1> WQ_hi:
00004A54 E461                <1>       IN    AL,SYS1                    ; wait for hi to lo
00004A56 A810                <1>       TEST  AL,010H                    ; transition on memory
00004A58 75FA                <1>       JNZ   SHORT WQ_hi         ; refresh.
00004A5A E461                <1> WQ_lo:  IN      AL,SYS1
00004A5C A810                <1>       TEST  AL,010H
00004A5E 74FA                <1>       JZ    SHORT WQ_lo
00004A60 E2ED                <1>       LOOP  WQ_1
                             <1>
00004A62 C605[D3580100]80    <1>         MOV     byte [DISK_STATUS1],TIME_OUT ; ERROR
00004A69 F9                  <1>       STC
                             <1> WQ_OK:
00004A6A C3                  <1>       RETn
                             <1> ;WQ_OK:     ;CLC
                             <1> ;     RETn
                             <1>
                             <1> ;---------------------------------------
                             <1> ;      CHECK FIXED DISK STATUS    :
                             <1> ;----------------------------------------
                             <1> CHECK_STATUS:
00004A6B E813000000          <1>        CALL   CHECK_ST             ; CHECK THE STATUS BYTE
00004A70 7509                <1>        JNZ    short CHECK_S1             ; AN ERROR WAS FOUND
00004A72 A801                <1>        TEST   AL,ST_ERROR          ; WERE THERE ANY OTHER ERRORS
00004A74 7405                <1>        JZ     short CHECK_S1             ; NO ERROR REPORTED
00004A76 E849000000          <1>        CALL   CHECK_ER             ; ERROR REPORTED
                             <1> CHECK_S1:
00004A7B 803D[D3580100]00    <1>        CMP    byte [DISK_STATUS1],0      ; SET STATUS FOR CALLER
00004A82 C3                  <1>        RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ;      CHECK FIXED DISK STATUS BYTE      :
                             <1> ;----------------------------------------
                             <1> CHECK_ST:
                             <1>        ;MOV   DX,HF_PORT+7         ; GET THE STATUS
00004A83 668B15[E85C0000]    <1>        mov    dx, [HF_PORT]
00004A8A 80C207              <1>        add    dl, 7
                             <1>
                             <1>        ; 17/02/2016
                             <1>        ;(http://wiki.osdev.org/ATA_PIO_Mode)
                             <1>        ;"delay 400ns to allow drive to set new values of BSY and DRQ"
00004A8D EC                  <1>        IN     AL,DX
                             <1>        ;in    al, dx ; 100ns
                             <1>        ;in    al, dx ; 100ns
                             <1>        ;in    al, dx ; 100ns
                             <1>        NEWIODELAY ; 18/02/2016 (AWARD BIOS - 1999, 'CKST' in AHSDK.ASM)
00004A8E E6EB                <2>  out 0ebh,al
                             <1>        ;
00004A90 A2[C9580100]        <1>        MOV    [HF_STATUS],AL
00004A95 B400                <1>        MOV    AH,0
00004A97 A880                <1>        TEST   AL,ST_BUSY           ; IF STILL BUSY
00004A99 751A                <1>        JNZ    short CKST_EXIT            ;  REPORT OK
00004A9B B4CC                <1>        MOV    AH,WRITE_FAULT
00004A9D A820                <1>        TEST   AL,ST_WRT_FLT        ; CHECK FOR WRITE FAULT
00004A9F 7514                <1>        JNZ    short CKST_EXIT
00004AA1 B4AA                <1>        MOV    AH,NOT_RDY
00004AA3 A840                <1>        TEST   AL,ST_READY          ; CHECK FOR NOT READY
00004AA5 740E                <1>        JZ     short CKST_EXIT
00004AA7 B440                <1>        MOV    AH,BAD_SEEK
00004AA9 A810                <1>        TEST   AL,ST_SEEK_COMPL     ; CHECK FOR SEEK NOT COMPLETE
00004AAB 7408                <1>        JZ     short CKST_EXIT
00004AAD B411                <1>        MOV    AH,DATA_CORRECTED
00004AAF A804                <1>        TEST   AL,ST_CORRCTD        ; CHECK FOR CORRECTED ECC
00004AB1 7502                <1>        JNZ    short CKST_EXIT
00004AB3 B400                <1>        MOV    AH,0
                             <1> CKST_EXIT:
00004AB5 8825[D3580100]      <1>        MOV    [DISK_STATUS1],AH    ; SET ERROR FLAG
00004ABB 80FC11              <1>        CMP    AH,DATA_CORRECTED    ; KEEP GOING WITH DATA CORRECTED
00004ABE 7403                <1>        JZ     short CKST_EX1
00004AC0 80FC00              <1>        CMP    AH,0
                             <1> CKST_EX1:
00004AC3 C3                  <1>        RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ;      CHECK FIXED DISK ERROR REGISTER :
                             <1> ;----------------------------------------
                             <1> CHECK_ER:
                             <1>        ;MOV   DX, HF_PORT+1        ; GET THE ERROR REGISTER
00004AC4 668B15[E85C0000]    <1>        mov    dx, [HF_PORT]        ;
00004ACB FEC2                <1>        inc    dl
00004ACD EC                  <1>        IN     AL,DX
00004ACE A2[CA580100]        <1>        MOV    [HF_ERROR],AL
00004AD3 53                  <1>        PUSH   eBX ; 21/02/2015
00004AD4 B908000000          <1>        MOV    eCX,8                ; TEST ALL 8 BITS
00004AD9 D0E0                <1> CK1:   SHL    AL,1                 ; MOVE NEXT ERROR BIT TO CARRY
00004ADB 7202                <1>        JC     short CK2            ; FOUND THE ERROR
00004ADD E2FA                <1>        LOOP   CK1                  ; KEEP TRYING
00004ADF BB[DC5C0000]        <1> CK2:   MOV    eBX, ERR_TBL         ; COMPUTE ADDRESS OF
00004AE4 01CB                <1>        ADD    eBX,eCX                    ; ERROR CODE
                             <1>        ;;MOV  AH,BYTE [CS:BX]            ; GET ERROR CODE
                             <1>        ;mov   ah, [bx]
00004AE6 8A23                <1>        mov    ah, [ebx] ; 21/02/2015
00004AE8 8825[D3580100]      <1> CKEX:  MOV    [DISK_STATUS1],AH    ; SAVE ERROR CODE
00004AEE 5B                  <1>        POP    eBX
00004AEF 80FC00              <1>        CMP    AH,0
00004AF2 C3                  <1>        RETn
                             <1>
                             <1> ;--------------------------------------------------------
                             <1> ; CHECK_DMA                                         :
                             <1> ;  -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL :
                             <1> ;   FIT WITHOUT SEGMENT OVERFLOW.                      :
                             <1> ;  -ES:BX HAS BEEN REVISED TO THE FORMAT SSSS:000X  :
                             <1> ;  -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE)      :
                             <1> ;  -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H) :
                             <1> ;  -ERROR OTHERWISE                                    :
                             <1> ;--------------------------------------------------------
                             <1> CHECK_DMA:
00004AF3 6650                <1>        PUSH   AX                   ; SAVE REGISTERS
00004AF5 66B80080            <1>        MOV    AX,8000H             ; AH = MAX # SECTORS AL = MAX OFFSET
00004AF9 F645FE02            <1>          TEST     byte [CMD_BLOCK+6],ECC_MODE
00004AFD 7404                <1>        JZ     short CKD1
00004AFF 66B8047F            <1>        MOV    AX,7F04H             ; ECC IS 4 MORE BYTES
00004B03 3A65F9              <1> CKD1:  CMP    AH, [CMD_BLOCK+1]    ; NUMBER OF SECTORS
00004B06 7706                <1>        JA     short CKDOK          ; IT WILL FIT
00004B08 7208                <1>        JB     short CKDERR         ; TOO MANY
00004B0A 38D8                <1>        CMP    AL,BL                ; CHECK OFFSET ON MAX SECTORS
00004B0C 7204                <1>        JB     short CKDERR         ; ERROR
00004B0E F8                  <1> CKDOK:        CLC                        ; CLEAR CARRY
00004B0F 6658                <1>        POP    AX
00004B11 C3                  <1>        RETn                        ; NORMAL RETURN
00004B12 F9                  <1> CKDERR: STC                        ; INDICATE ERROR
```

```
00004B13 C605[D3580100]09    <1>         MOV     byte [DISK_STATUS1],DMA_BOUNDARY
00004B1A 6658                <1>       POP    AX
00004B1C C3                  <1>       RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     SET UP ES:BX-> DISK PARMS :
                             <1> ;----------------------------------------
                             <1>
                             <1> ; INPUT -> DL = 0 based drive number
                             <1> ; OUTPUT -> ES:BX = disk parameter table address
                             <1>
                             <1> GET_VEC:
                             <1>       ;SUB   AX,AX               ; GET DISK PARAMETER ADDRESS
                             <1>       ;MOV   ES,AX
                             <1>       ;TEST  DL,1
                             <1>       ;JZ    short GV_0
                             <1> ;     LES    BX,[HF1_TBL_VEC]    ; ES:BX -> DRIVE PARAMETERS
                             <1> ;     JMP    SHORT GV_EXIT
                             <1> ;GV_0:
                             <1> ;     LES    BX,[HF_TBL_VEC]           ; ES:BX -> DRIVE PARAMETERS
                             <1> ;
                             <1>       ;xor   bh, bh
00004B1D 31DB                <1>       xor    ebx, ebx
00004B1F 88D3                <1>       mov    bl, dl
                             <1>       ;;02/01/2015
                             <1>       ;;shl  bl, 1               ; port address offset
                             <1>       ;;mov  ax, [bx+hd_ports]   ; Base port address (1F0h, 170h)
                             <1>       ;;shl  bl, 1               ; dpt pointer offset
00004B21 C0E302              <1>       shl    bl, 2  ;;
                             <1>       ;add   bx, HF_TBL_VEC            ; Disk parameter table pointer
00004B24 81C3[D8580100]      <1>       add    ebx, HF_TBL_VEC ; 21/02/2015
                             <1>       ;push  word [bx+2]         ; dpt segment
                             <1>       ;pop   es
                             <1>       ;mov   bx, [bx]            ; dpt offset
00004B2A 8B1B                <1>       mov    ebx, [ebx]
                             <1> ;GV_EXIT:
00004B2C C3                  <1>       RETn
                             <1>
                             <1> hdc1_int: ; 21/02/2015
                             <1> ;--- HARDWARE INT 76H -- ( IRQ LEVEL  14 ) ----------------------
                             <1> ;                                                  :
                             <1> ;     FIXED DISK INTERRUPT ROUTINE                       :
                             <1> ;                                                  :
                             <1> ;----------------------------------------------------------------
                             <1>
                             <1> ; 22/12/2014
                             <1> ; IBM PC-XT Model 286 System BIOS Source Code - DISK.ASM (HD_INT)
                             <1> ;      '11/15/85'
                             <1> ; AWARD BIOS 1999 (D1A0622)
                             <1> ;     Source Code - ATORGS.ASM (INT_HDISK, INT_HDISK1)
                             <1>
                             <1> ;int_76h:
                             <1> HD_INT:
00004B2D 6650                <1>       PUSH   AX
00004B2F 1E                  <1>       PUSH   DS
                             <1>       ;CALL  DDS
                             <1>       ; 21/02/2015 (32 bit, 386 pm modification)
00004B30 66B81000            <1>       mov    ax, KDATA
00004B34 8ED8                <1>       mov    ds, ax
                             <1>       ;
                             <1>       ;;MOV  @HF_INT_FLAG,0FFH   ; ALL DONE
                             <1>         ;mov     byte [CS:HF_INT_FLAG], 0FFh
00004B36 C605[CB580100]FF    <1>       mov    byte [HF_INT_FLAG], 0FFh
                             <1>       ;
00004B3D 6652                <1>       push   dx
00004B3F 66BAF701            <1>       mov    dx, HDC1_BASEPORT+7 ; Status Register (1F7h)
                             <1>                                  ; Clear Controller
                             <1> Clear_IRQ1415:                         ; (Award BIOS - 1999)
00004B43 EC                  <1>       in     al, dx              ;
00004B44 665A                <1>       pop    dx
                             <1>       NEWIODELAY
00004B46 E6EB                <2>  out 0ebh,al
                             <1>       ;
00004B48 B020                <1>       MOV    AL,EOI              ; NON-SPECIFIC END OF INTERRUPT
00004B4A E6A0                <1>       OUT    INTB00,AL           ; FOR CONTROLLER #2
                             <1>       ;JMP   $+2                 ; WAIT
                             <1>       NEWIODELAY
00004B4C E6EB                <2>  out 0ebh,al
00004B4E E620                <1>       OUT    INTA00,AL           ; FOR CONTROLLER #1
00004B50 1F                  <1>       POP    DS
                             <1>       ;STI                       ; RE-ENABLE INTERRUPTS
                             <1>       ;MOV   AX,9100H            ; DEVICE POST
                             <1>       ;INT   15H                 ;  INTERRUPT
                             <1> irq15_iret: ; 25/02/2015
00004B51 6658                <1>       POP    AX
00004B53 CF                  <1>       IRETd                      ; RETURN FROM INTERRUPT
                             <1>
                             <1> hdc2_int: ; 21/02/2015
                             <1> ;++++ HARDWARE INT 77H ++ ( IRQ LEVEL  15 ) +++++++++++++++++++++
                             <1> ;                                                  :
                             <1> ;     FIXED DISK INTERRUPT ROUTINE                       :
                             <1> ;                                                  :
                             <1> ;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                             <1>
                             <1> ;int_77h:
                             <1> HD1_INT:
00004B54 6650                <1>       PUSH   AX
                             <1>       ; Check if that is a spurious IRQ (from slave PIC)
                             <1>       ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
00004B56 B00B                <1>       mov    al, 0Bh  ; In-Service Register
00004B58 E6A0                <1>       out    0A0h, al
00004B5A EB00                <1>         jmp short $+2
00004B5C EB00                <1>       jmp short $+2
```

```
00004B5E E4A0                   <1>       in     al, 0A0h
00004B60 2480                   <1>       and    al, 80h ; bit 7 (is it real IRQ 15 or fake?)
00004B62 74ED                   <1>       jz     short irq15_iret ; Fake (spurious)IRQ, do not send EOI)
                                <1>       ;
00004B64 1E                     <1>       PUSH   DS
                                <1>       ;CALL  DDS
                                <1>       ; 21/02/2015 (32 bit, 386 pm modification)
00004B65 66B81000               <1>       mov    ax, KDATA
00004B69 8ED8                   <1>       mov    ds, ax
                                <1>       ;
                                <1>       ;;MOV  @HF_INT_FLAG,0FFH   ; ALL DONE
                                <1>         ;or      byte [CS:HF_INT_FLAG],0C0h
00004B6B 800D[CB580100]C0       <1>       or     byte [HF_INT_FLAG], 0C0h
                                <1>       ;
00004B72 6652                   <1>       push   dx
00004B74 66BA7701               <1>       mov    dx, HDC2_BASEPORT+7 ; Status Register (177h)
                                <1>                                ; Clear Controller (Award BIOS 1999)
00004B78 EBC9                   <1>       jmp    short Clear_IRQ1415
                                <1>
                                <1>
                                <1> ;%include 'diskdata.inc' ; 11/03/2015
                                <1> ;%include 'diskbss.inc' ; 11/03/2015
                                <1>
                                <1>
                                <1> ;////////////////////////////////////////////////////////////////////
                                <1> ;; END OF DISK I/O SYTEM ///
                                    %include 'memory.s'  ; 09/03/2015
                                <1> ; ****************************************************************************
                                <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - memory.s
                                <1> ; ----------------------------------------------------------------------------
                                <1> ; Last Update: 22/07/2017
                                <1> ; ----------------------------------------------------------------------------
                                <1> ; Beginning: 24/01/2016
                                <1> ; ----------------------------------------------------------------------------
                                <1> ; Assembler: NASM version 2.11 (trdos386.s)
                                <1> ; ----------------------------------------------------------------------------
                                <1> ; Turkish Rational DOS
                                <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
                                <1> ;
                                <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                                <1> ; memory.inc (18/10/2015)
                                <1> ; ****************************************************************************
                                <1>
                                <1> ; MEMORY.ASM - Retro UNIX 386 v1 MEMORY MANAGEMENT FUNCTIONS (PROCEDURES)
                                <1> ; Retro UNIX 386 v1 Kernel (unix386.s, v0.2.0.14) - MEMORY.INC
                                <1> ; Last Modification: 18/10/2015
                                <1>
                                <1> ; ///////// MEMORY MANAGEMENT FUNCTIONS (PROCEDURES) ///////////////
                                <1>
                                <1> ;;04/11/2014 (unix386.s)
                                <1> ;PDE_A_PRESENT     equ    1            ; Present flag for PDE
                                <1> ;PDE_A_WRITE equ    2            ; Writable (write permission) flag
                                <1> ;PDE_A_USER  equ    4            ; User (non-system/kernel) page flag
                                <1> ;;
                                <1> ;PTE_A_PRESENT     equ    1            ; Present flag for PTE (bit 0)
                                <1> ;PTE_A_WRITE equ    2            ; Writable (write permission) flag (bit 1)
                                <1> ;PTE_A_USER  equ    4            ; User (non-system/kernel) page flag (bit 2)
                                <1> ;PTE_A_ACCESS   equ      32          ; Accessed flag (bit 5) ; 09/03/2015
                                <1>
                                <1> ; 27/04/2015
                                <1> ; 09/03/2015
                                <1> PAGE_SIZE    equ 4096            ; page size in bytes
                                <1> PAGE_SHIFT   equ 12              ; page table shift count
                                <1> PAGE_D_SHIFT       equ 22 ; 12 + 10    ; page directory shift count
                                <1> PAGE_OFF     equ 0FFFh           ; 12 bit byte offset in page frame
                                <1> PTE_MASK     equ 03FFh           ; page table entry mask
                                <1> PTE_DUPLICATED  equ 200h         ; duplicated page sign (AVL bit 0)
                                <1> PDE_A_CLEAR  equ 0F000h          ; to clear PDE attribute bits
                                <1> PTE_A_CLEAR  equ 0F000h          ; to clear PTE attribute bits
                                <1> LOGIC_SECT_SIZE equ 512                ; logical sector size
                                <1> ERR_MAJOR_PF equ 0E0h            ; major error: page fault
                                <1> ERR_MINOR_IM equ 4 ;15/10/2016 (1->4); insufficient (out of) memory
                                <1> ERR_MINOR_PV equ 6 ;15/10/2016 (1->4); protection violation
                                <1> SWP_DISK_READ_ERR      equ 40
                                <1> SWP_DISK_NOT_PRESENT_ERR    equ 41
                                <1> SWP_SECTOR_NOT_PRESENT_ERR equ 42
                                <1> SWP_NO_FREE_SPACE_ERR      equ 43
                                <1> SWP_DISK_WRITE_ERR         equ 44
                                <1> SWP_NO_PAGE_TO_SWAP_ERR    equ 45
                                <1> PTE_A_ACCESS_BIT equ 5  ; Bit 5 (accessed flag)
                                <1> SECTOR_SHIFT     equ 3  ; sector shift (to convert page block number)
                                <1> ; 12/07/2016
                                <1> PTE_SHARED    equ 400h           ; AVL bit 1, direct memory access bit
                                <1>                                  ; (Indicates that the page is not allocated
                                <1>                                  ; for the process, it is a shared or system
                                <1>                                         ; page, it must not be deallocated!)
                                <1> ;
                                <1> ;; Retro Unix 386 v1 - paging method/principles
                                <1> ;;
                                <1> ;; 10/10/2014
                                <1> ;; RETRO UNIX 386 v1 - PAGING METHOD/PRINCIPLES
                                <1> ;;
                                <1> ;; KERNEL PAGE MAP: 1 to 1 physical memory page map
                                <1> ;;    (virtual address = physical address)
                                <1> ;; KERNEL PAGE TABLES:
                                <1> ;;    Kernel page directory and all page tables are
                                <1> ;;    on memory as initialized, as equal to physical memory
                                <1> ;;    layout. Kernel pages can/must not be swapped out/in.
                                <1> ;;
                                <1> ;;    what for: User pages may be swapped out, when accessing
                                <1> ;;    a page in kernel/system mode, if it would be swapped out,
                                <1> ;;    kernel would have to swap it in! But it is also may be
                                <1> ;;    in use by a user process. (In system/kernel mode
<1> ;;    kernel can access all memory pages even if they are
<1> ;;    reserved/allocated for user processes. Swap out/in would
<1> ;;    cause conflicts.)
<1> ;;
<1> ;;    As result of these conditions,
<1> ;;    all kernel pages must be initialized as equal to
<1> ;;    physical layout for preventing page faults.
<1> ;;    Also, calling "allocate page" procedure after
<1> ;;    a page fault can cause another page fault (double fault)
<1> ;;    if all kernel page tables would not be initialized.
<1> ;;
<1> ;;    [first_page] = Beginning of users space, as offset to
<1> ;;    memory allocation table. (double word aligned)
<1> ;;
<1> ;;    [next_page] = first/next free space to be searched
<1> ;;    as offset to memory allocation table. (dw aligned)
<1> ;;
<1> ;;    [last_page] = End of memory (users space), as offset
<1> ;;    to memory allocation table. (double word aligned)
<1> ;;
<1> ;; USER PAGE TABLES:
<1> ;;    Demand paging (& 'copy on write' allocation method) ...
<1> ;;           'ready only' marked copies of the
<1> ;;           parent process's page table entries (for
<1> ;;           same physical memory).
<1> ;;           (A page will be copied to a new page after
<1> ;;            if it causes R/W page fault.)
<1> ;;
<1> ;;    Every user process has own (different)
<1> ;;    page directory and page tables.
<1> ;;
<1> ;;    Code starts at virtual address 0, always.
<1> ;;    (Initial value of EIP is 0 in user mode.)
<1> ;;    (Programs can be written/developed as simple
<1> ;;     flat memory programs.)
<1> ;;
<1> ;; MEMORY ALLOCATION STRATEGY:
<1> ;;    Memory page will be allocated by kernel only
<1> ;;           (in kernel/system mode only).
<1> ;;    * After a
<1> ;;      - 'not present' page fault
<1> ;;      - 'writing attempt on read only page' page fault
<1> ;;    * For loading (opening, reading) a file or disk/drive
<1> ;;    * As responce to 'allocate additional memory blocks'
<1> ;;      request by running process.
<1> ;;    * While creating a process, allocating a new buffer,
<1> ;;      new page tables etc.
<1> ;;
<1> ;;    At first,
<1> ;;    - 'allocate page' procedure will be called;
<1> ;;       if it will return with a valid (>0) physical address
<1> ;;       (that means the relevant M.A.T. bit has been RESET)
<1> ;;       relevant memory page/block will be cleared (zeroed).
<1> ;;    - 'allocate page' will be called for allocating page
<1> ;;      directory, page table and running space (data/code).
<1> ;;    - every successful 'allocate page' call will decrease
<1> ;;      'free_pages' count (pointer).
<1> ;;    - 'out of (insufficient) memory error' will be returned
<1> ;;      if 'free_pages' points to a ZERO.
<1> ;;    - swapping out and swapping in (if it is not a new page)
<1> ;;      procedures will be called as responce to 'out of memory'
<1> ;;      error except errors caused by attribute conflicts.
<1> ;;     (swapper functions)
<1> ;;
<1> ;;    At second,
<1> ;;    - page directory entry will be updated then page table
<1> ;;      entry will be updated.
<1> ;;
<1> ;; MEMORY ALLOCATION TABLE FORMAT:
<1> ;;    - M.A.T. has a size according to available memory as
<1> ;;      follows:
<1> ;;             - 1 (allocation) bit per 1 page (4096 bytes)
<1> ;;             - a bit with value of 0 means allocated page
<1> ;;             - a bit with value of 1 means a free page
<1> ;;    - 'free_pages' pointer holds count of free pages
<1> ;;      depending on M.A.T.
<1> ;;           (NOTE: Free page count will not be checked
<1> ;;           again -on M.A.T.- after initialization.
<1> ;;           Kernel will trust on initial count.)
<1> ;;    - 'free_pages' count will be decreased by allocation
<1> ;;      and it will be increased by deallocation procedures.
<1> ;;
<1> ;;    - Available memory will be calculated during
<1> ;;      the kernel's initialization stage (in real mode).
<1> ;;      Memory allocation table and kernel page tables
<1> ;;      will be formatted/sized as result of available
<1> ;;      memory calculation before paging is enabled.
<1> ;;
<1> ;; For 4GB Available/Present Memory: (max. possible memory size)
<1> ;;    - Memory Allocation Table size will be 128 KB.
<1> ;;    - Memory allocation for kernel page directory size
<1> ;;      is always 4 KB. (in addition to total allocation size
<1> ;;      for page tables)
<1> ;;    - Memory allocation for kernel page tables (1024 tables)
<1> ;;      is 4 MB (1024*4*1024 bytes).
<1> ;;    - User (available) space will be started
<1> ;;      at 6th MB of the memory (after 1MB+4MB).
<1> ;;    - The first 640 KB is for kernel's itself plus
<1> ;;      memory allocation table and kernel's page directory
<1> ;;      (D0000h-EFFFFh may be used as kernel space...)
<1> ;;    - B0000h to B7FFFh address space (32 KB) will be used
<1> ;;      for buffers.
<1> ;;    - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.
```

```
<1> ;;      (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)
<1> ;;    - Kernel page tables start at 100000h (2nd MB)
<1> ;;
<1> ;; For 1GB Available Memory:
<1> ;;    - Memory Allocation Table size will be 32 KB.
<1> ;;    - Memory allocation for kernel page directory size
<1> ;;      is always 4 KB. (in addition to total allocation size
<1> ;;      for page tables)
<1> ;;    - Memory allocation for kernel page tables (256 tables)
<1> ;;      is 1 MB (256*4*1024 bytes).
<1> ;;    - User (available) space will be started
<1> ;;      at 3th MB of the memory (after 1MB+1MB).
<1> ;;    - The first 640 KB is for kernel's itself plus
<1> ;;      memory allocation table and kernel's page directory
<1> ;;      (D0000h-EFFFFh may be used as kernel space...)
<1> ;;    - B0000h to B7FFFh address space (32 KB) will be used
<1> ;;      for buffers.
<1> ;;    - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.
<1> ;;      (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)
<1> ;;    - Kernel page tables start at 100000h (2nd MB).
<1> ;;
<1> ;;
<1>
<1>
<1> ;;************************************************************************************
<1> ;;
<1> ;; RETRO UNIX 386 v1 - Paging (Method for Copy On Write paging principle)
<1> ;; DEMAND PAGING - PARENT&CHILD PAGE TABLE DUPLICATION PRINCIPLES (23/04/2015)
<1>
<1> ;; Main factor: "sys fork" system call
<1> ;;
<1> ;;          FORK
<1> ;;                      |----> parent - duplicated PTEs, read only pages
<1> ;;  writable pages ---->|
<1> ;;                      |----> child - duplicated PTEs, read only pages
<1> ;;
<1> ;; AVL bit (0) of Page Table Entry is used as duplication sign
<1> ;;
<1> ;; AVL Bit 0 [PTE Bit 9] = 'Duplicated PTE belongs to child' sign/flag (if it is set)
<1> ;; Note: Dirty bit (PTE bit 6) may be used instead of AVL bit 0 (PTE bit 9)
<1> ;;       -while R/W bit is 0-.
<1> ;;
<1> ;; Duplicate page tables with writable pages (the 1st sys fork in the process):
<1> ;; # Parent's Page Table Entries are updated to point same pages as read only,
<1> ;;   as duplicated PTE bit  -AVL bit 0, PTE bit 9- are reset/clear.
<1> ;; # Then Parent's Page Table is copied to Child's Page Table.
<1> ;; # Child's Page Table Entries are updated as duplicated child bit
<1> ;;   -AVL bit 0, PTE bit 9- is set.
<1> ;;
<1> ;; Duplicate page tables with read only pages (several sys fork system calls):
<1> ;; # Parent's read only pages are copied to new child pages.
<1> ;;   Parent's PTE attributes are not changed.
<1> ;;   (Because, there is another parent-child fork before this fork! We must not
<1> ;;    destroy/mix previous fork result).
<1> ;; # Child's Page Table Entries (which are corresponding to Parent's
<1> ;;   read only pages) are set as writable (while duplicated PTE bit is clear).
<1> ;; # Parent's PTEs with writable page attribute are updated to point same pages
<1> ;;   as read only, (while) duplicated PTE bit is reset (clear).
<1> ;; # Parent's Page Table Entries (with writable page attribute) are duplicated
<1> ;;   as Child's Page Table Entries without copying actual page.
<1> ;; # Child 's Page Table Entries (which are corresponding to Parent's writable
<1> ;;   pages) are updated as duplicated PTE bit (AVL bit 0, PTE bit 9- is set.
<1> ;;
<1> ;; !? WHAT FOR (duplication after duplication):
<1> ;; In UNIX method for sys fork (a typical 'fork' application in /etc/init)
<1> ;; program/executable code continues from specified location as child process,
<1> ;; returns back previous code location as parent process, every child after
<1> ;; every sys fork uses last image of code and data just prior the fork.
<1> ;; Even if the parent code changes data, the child will not see the changed data
<1> ;; after the fork. In Retro UNIX 8086 v1, parent's process segment (32KB)
<1> ;; was copied to child's process segment (all of code and data) according to
<1> ;; original UNIX v1 which copies all of parent process code and data -core-
<1> ;; to child space -core- but swaps that core image -of child- on to disk.
<1> ;; If I (Erdogan Tan) would use a method of to copy parent's core
<1> ;; (complete running image of parent process) to the child process;
<1> ;; for big sizes, i would force Retro UNIX 386 v1 to spend many memory pages
<1> ;; and times only for a sys fork. (It would excessive reservation for sys fork,
<1> ;; because sys fork usually is prior to sys exec; sys exec always establishes
<1> ;; a new/fresh core -running space-, by clearing all code/data content).
<1> ;; 'Read Only' page flag ensures page fault handler is needed only for a few write
<1> ;; attempts between sys fork and sys exec, not more... (I say so by thinking
<1> ;; of "/etc/init" content, specially.) sys exec will clear page tables and
<1> ;; new/fresh pages will be used to load and run new executable/program.
<1> ;; That is what for i have preferred "copy on write", "duplication" method
<1> ;; for sharing same read only pages between parent and child processes.
<1> ;; That is a pitty i have to use new private flag (AVL bit 0, "duplicated PTE
<1> ;; belongs to child" sign) for cooperation on duplicated pages between a parent
<1> ;; and it's child processes; otherwise parent process would destroy data belongs
<1> ;; to its child or vice versa; or some pages would remain unclaimed
<1> ;; -deallocation problem-.
<1> ;; Note: to prevent conflicts, read only pages must not be swapped out...
<1> ;;
<1> ;; WHEN PARENT TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:
<1> ;; # Page fault handler will do those:
<1> ;;   - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).
<1> ;;   - If it is reset/clear, there is a child uses same page.
<1> ;;   - Parent's read only page -previous page- is copied to a new writable page.
<1> ;;   - Parent's PTE is updated as writable page, as unique page (AVL=0)
<1> ;;   - (Page fault handler whill check this PTE later, if child process causes to
<1> ;;     page fault due to write attempt on read only page. Of course, the previous
<1> ;;     read only page will be converted to writable and unique page which belongs
<1> ;;     to child process.)
<1> ;; WHEN CHILD TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:
```

```
                             <1> ;; # Page fault handler will do those:
                             <1> ;;   - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).
                             <1> ;;   - If it is set, there is a parent uses -or was using- same page.
                             <1> ;;   - Same PTE address within parent's page table is checked if it has same page
                             <1> ;;     address or not.
                             <1> ;;   - If parent's PTE has same address, child will continue with a new writable page.
                             <1> ;;     Parent's PTE will point to same (previous) page as writable, unique (AVL=0).
                             <1> ;;   - If parent's PTE has different address, child will continue with it's
                             <1> ;;     own/same page but read only flag (0) will be changed to writable flag (1) and
                             <1> ;;     'duplicated PTE (belongs to child)' flag/sign will be cleared/reset.
                             <1> ;;
                             <1> ;; NOTE: When a child process is terminated, read only flags of parent's page tables
                             <1> ;;       will be set as writable (and unique) in case of child process was using
                             <1> ;;       same pages with duplicated child PTE sign... Depending on sys fork and
                             <1> ;;       duplication method details, it is not possible multiple child processes
                             <1> ;;       were using same page with duplicated PTEs.
                             <1> ;;
                             <1> ;;**************************************************************************************
                             <1>
                             <1> ;; 08/10/2014
                             <1> ;; 11/09/2014 - Retro UNIX 386 v1 PAGING (further) draft
                             <1> ;;          by Erdogan Tan (Based on KolibriOS 'memory.inc')
                             <1>
                             <1> ;; 'allocate_page' code is derived and modified from KolibriOS
                             <1> ;; 'alloc_page' procedure in 'memory.inc'
                             <1> ;; (25/08/2014, Revision: 5057) file
                             <1> ;; by KolibriOS Team (2004-2012)
                             <1>
                             <1> allocate_page:
                             <1>       ; 01/07/2015
                             <1>       ; 05/05/2015
                             <1>       ; 30/04/2015
                             <1>       ; 16/10/2014
                             <1>       ; 08/10/2014
                             <1>       ; 09/09/2014 (Retro UNIX 386 v1 - beginning)
                             <1>       ;
                             <1>       ; INPUT -> none
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
                             <1>       ;      (corresponding MEMORY ALLOCATION TABLE bit is RESET)
                             <1>       ;
                             <1>       ;      CF = 1 and EAX = 0
                             <1>       ;                if there is not a free page to be allocated
                             <1>       ;
                             <1>       ; Modified Registers -> none (except EAX)
                             <1>       ;
00004B7A A1[40580100]        <1>       mov    eax, [free_pages]
00004B7F 21C0                <1>       and    eax, eax
00004B81 7438                <1>       jz     short out_of_memory
                             <1>       ;
00004B83 53                  <1>       push   ebx
00004B84 51                  <1>       push   ecx
                             <1>       ;
00004B85 BB00001000          <1>       mov    ebx, MEM_ALLOC_TBL   ; Memory Allocation Table offset
00004B8A 89D9                <1>       mov    ecx, ebx
                             <1>                                  ; NOTE: 32 (first_page) is initial
                             <1>                                  ; value of [next_page].
                             <1>                                  ; It points to the first available
                             <1>                                  ; page block for users (ring 3) ...
                             <1>                                  ; (MAT offset 32 = 1024/32)
                             <1>                                  ; (at the of the first 4 MB)
00004B8C 031D[44580100]      <1>       add    ebx, [next_page] ; Free page searching starts from here
                             <1>                              ; next_free_page >> 5
00004B92 030D[48580100]      <1>       add    ecx, [last_page] ; Free page searching ends here
                             <1>                              ; (total_pages - 1) >> 5
                             <1> al_p_scan:
00004B98 39CB                <1>       cmp    ebx, ecx
00004B9A 770A                <1>       ja     short al_p_notfound
                             <1>       ;
                             <1>       ; 01/07/2015
                             <1>       ; AMD64 Architecture Programmer’s Manual
                             <1>       ; Volume 3:
                             <1>       ; General-Purpose and System Instructions
                             <1>       ;
                             <1>       ; BSF - Bit Scan Forward
                             <1>       ;
                             <1>       ;   Searches the value in a register or a memory location
                             <1>       ;   (second operand) for the least-significant set bit.
                             <1>       ;   If a set bit is found, the instruction clears the zero flag (ZF)
                             <1>       ;   and stores the index of the least-significant set bit in a destination
                             <1>       ;   register (first operand). If the second operand contains 0,
                             <1>       ;   the instruction sets ZF to 1 and does not change the contents of the
                             <1>       ;   destination register. The bit index is an unsigned offset from bit 0
                             <1>       ;   of the searched value
                             <1>       ;
00004B9C 0FBC03              <1>       bsf    eax, [ebx] ; Scans source operand for first bit set (1).
                             <1>                         ; Clear ZF if a bit is found set (1) and
                             <1>                         ; loads the destination with an index to
                             <1>                         ; first set bit. (0 -> 31)
                             <1>                         ; Sets ZF to 1 if no bits are found set.
00004B9F 7525                <1>       jnz    short al_p_found ; ZF = 0 -> a free page has been found
                             <1>                       ;
                             <1>                       ; NOTE:  a Memory Allocation Table bit
                             <1>                       ;       with value of 1 means
                             <1>                       ;       the corresponding page is free
                             <1>                       ;       (Retro UNIX 386 v1 feature only!)
00004BA1 83C304              <1>       add    ebx, 4
                             <1>                       ; We return back for searching next page block
                             <1>                       ; NOTE: [free_pages] is not ZERO; so,
                             <1>                       ;      we always will find at least 1 free page here.
00004BA4 EBF2                <1>         jmp     short al_p_scan
                             <1>       ;
                             <1> al_p_notfound:
00004BA6 81E900001000        <1>       sub    ecx, MEM_ALLOC_TBL
00004BAC 890D[44580100]      <1>       mov    [next_page], ecx ; next/first free page = last page
                             <1>                            ; (deallocate_page procedure will change it)
00004BB2 31C0                <1>       xor    eax, eax
00004BB4 A3[40580100]        <1>       mov    [free_pages], eax ; 0
00004BB9 59                  <1>       pop    ecx
00004BBA 5B                  <1>       pop    ebx
                             <1>       ;
                             <1> out_of_memory:
00004BBB E85B040000          <1>       call   swap_out
00004BC0 7325                <1>       jnc    short al_p_ok  ; [free_pages] = 0, re-allocation by swap_out
                             <1>       ;
00004BC2 29C0                <1>       sub    eax, eax ; 0
00004BC4 F9                  <1>       stc
00004BC5 C3                  <1>       retn
                             <1>
                             <1> al_p_found:
00004BC6 89D9                <1>       mov    ecx, ebx
00004BC8 81E900001000        <1>       sub    ecx, MEM_ALLOC_TBL
00004BCE 890D[44580100]      <1>       mov    [next_page], ecx ; Set first free page searching start
                             <1>                            ; address/offset (to the next)
00004BD4 FF0D[40580100]      <1>         dec     dword [free_pages] ; 1 page has been allocated (X = X-1)
                             <1>       ;
00004BDA 0FB303              <1>       btr    [ebx], eax    ; The destination bit indexed by the source value
                             <1>                            ; is copied into the Carry Flag and then cleared
                             <1>                            ; in the destination.
                             <1>                            ;
                             <1>                            ; Reset the bit which is corresponding to the
                             <1>                            ; (just) allocated page.
                             <1>       ; 01/07/2015 (4*8 = 32, 1 allocation byte = 8 pages)
00004BDD C1E103              <1>       shl    ecx, 3        ; (page block offset * 32) + page index
00004BE0 01C8                <1>       add    eax, ecx      ; = page number
00004BE2 C1E00C              <1>       shl    eax, 12             ; physical address of the page (flat/real value)
                             <1>       ; EAX = physical address of memory page
                             <1>       ;
                             <1>       ; NOTE: The relevant page directory and page table entry will be updated
                             <1>       ;       according to this EAX value...
00004BE5 59                  <1>       pop    ecx
00004BE6 5B                  <1>       pop    ebx
                             <1> al_p_ok:
00004BE7 C3                  <1>       retn
                             <1>
                             <1>
                             <1> make_page_dir:
                             <1>       ; 18/04/2015
                             <1>       ; 12/04/2015
                             <1>       ; 23/10/2014
                             <1>       ; 16/10/2014
                             <1>       ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      none
                             <1>       ; OUTPUT ->
                             <1>       ;      (EAX = 0)
                             <1>       ;      cf = 1 -> insufficient (out of) memory error
                             <1>       ;      cf = 0 ->
                             <1>       ;      u.pgdir = page directory (physical) address of the current
                             <1>       ;              process/user.
                             <1>       ;
                             <1>       ; Modified Registers -> EAX
                             <1>       ;
00004BE8 E88DFFFFFF          <1>       call   allocate_page
00004BED 7216                <1>       jc     short mkpd_error
                             <1>       ;
00004BEF A3[B8030300]        <1>       mov    [u.pgdir], eax    ; Page dir address for current user/process
                             <1>                             ; (Physical address)
                             <1> clear_page:
                             <1>       ; 18/04/2015
                             <1>       ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      EAX = physical address of the page
                             <1>       ; OUTPUT ->
                             <1>       ;      all bytes of the page will be cleared
                             <1>       ;
                             <1>       ; Modified Registers -> none
                             <1>       ;
00004BF4 57                  <1>       push   edi
00004BF5 51                  <1>       push   ecx
00004BF6 50                  <1>       push   eax
00004BF7 B900040000          <1>       mov    ecx, PAGE_SIZE / 4
00004BFC 89C7                <1>       mov    edi, eax
00004BFE 31C0                <1>       xor    eax, eax
00004C00 F3AB                <1>       rep    stosd
00004C02 58                  <1>       pop    eax
00004C03 59                  <1>       pop    ecx
00004C04 5F                  <1>       pop    edi
                             <1> mkpd_error:
                             <1> mkpt_error:
00004C05 C3                  <1>       retn
                             <1>
                             <1> make_page_table:
                             <1>       ; 23/06/2015
                             <1>       ; 18/04/2015
                             <1>       ; 12/04/2015
                             <1>       ; 16/10/2014
                             <1>       ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      EBX = virtual (linear) address
                             <1>       ;      ECX = page table attributes (lower 12 bits)
                             <1>       ;            (higher 20 bits must be ZERO)
```

```
                              <1>       ;            (bit 0 must be 1)
                              <1>       ;      u.pgdir = page directory (physical) address
                              <1>       ; OUTPUT ->
                              <1>       ;      EDX = Page directory entry address
                              <1>       ;      EAX = Page table address
                              <1>       ;      cf = 1 -> insufficient (out of) memory error
                              <1>       ;      cf = 0 -> page table address in the PDE (EDX)
                              <1>       ;
                              <1>       ; Modified Registers -> EAX, EDX
                              <1>       ;
00004C06 E86FFFFFFF           <1>       call   allocate_page
00004C0B 72F8                 <1>       jc     short mkpt_error
00004C0D E811000000           <1>       call   set_pde
00004C12 EBE0                 <1>       jmp    short clear_page
                              <1>
                              <1> make_page:
                              <1>       ; 24/07/2015
                              <1>       ; 23/06/2015 ; (Retro UNIX 386 v1 - beginning)
                              <1>       ;
                              <1>       ; INPUT ->
                              <1>       ;      EBX = virtual (linear) address
                              <1>       ;      ECX = page attributes (lower 12 bits)
                              <1>       ;            (higher 20 bits must be ZERO)
                              <1>       ;            (bit 0 must be 1)
                              <1>       ;      u.pgdir = page directory (physical) address
                              <1>       ; OUTPUT ->
                              <1>       ;      EBX = Virtual address
                              <1>       ;      (EDX = PTE value)
                              <1>       ;      EAX = Physical address
                              <1>       ;      cf = 1 -> insufficient (out of) memory error
                              <1>       ;
                              <1>       ; Modified Registers -> EAX, EDX
                              <1>       ;
00004C14 E861FFFFFF           <1>       call   allocate_page
00004C19 7207                 <1>       jc     short mkp_err
00004C1B E821000000           <1>       call   set_pte
00004C20 73D2                 <1>       jnc    short clear_page ; 18/04/2015
                              <1> mkp_err:
00004C22 C3                   <1>       retn
                              <1>
                              <1>
                              <1> set_pde:     ; Set page directory entry (PDE)
                              <1>       ; 20/07/2015
                              <1>       ; 18/04/2015
                              <1>       ; 12/04/2015
                              <1>       ; 23/10/2014
                              <1>       ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
                              <1>       ;
                              <1>       ; INPUT ->
                              <1>       ;      EAX = physical address
                              <1>       ;            (use present value if EAX = 0)
                              <1>       ;      EBX = virtual (linear) address
                              <1>       ;      ECX = page table attributes (lower 12 bits)
                              <1>       ;            (higher 20 bits must be ZERO)
                              <1>       ;            (bit 0 must be 1)
                              <1>       ;      u.pgdir = page directory (physical) address
                              <1>       ; OUTPUT ->
                              <1>       ;      EDX = PDE address
                              <1>       ;      EAX = page table address (physical)
                              <1>       ;      ;(CF=1 -> Invalid page address)
                              <1>       ;
                              <1>       ; Modified Registers -> EDX
                              <1>       ;
00004C23 89DA                 <1>       mov    edx, ebx
00004C25 C1EA16               <1>       shr    edx, PAGE_D_SHIFT ; 22
00004C28 C1E202               <1>       shl    edx, 2 ; offset to page directory (1024*4)
00004C2B 0315[B8030300]       <1>       add    edx, [u.pgdir]
                              <1>       ;
00004C31 21C0                 <1>       and    eax, eax
00004C33 7506                 <1>       jnz    short spde_1
                              <1>       ;
00004C35 8B02                 <1>       mov    eax, [edx]  ; old PDE value
                              <1>       ;test  al, 1
                              <1>       ;jz    short spde_2
00004C37 662500F0             <1>       and    ax, PDE_A_CLEAR ; 0F000h  ; clear lower 12 bits
                              <1> spde_1:
                              <1>       ;and   cx, 0FFFh
00004C3B 8902                 <1>       mov    [edx], eax
00004C3D 66090A               <1>       or     [edx], cx
00004C40 C3                   <1>       retn
                              <1> ;spde_2: ; error
                              <1> ;     stc
                              <1> ;     retn
                              <1>
                              <1> set_pte:     ; Set page table entry (PTE)
                              <1>       ; 24/07/2015
                              <1>       ; 20/07/2015
                              <1>       ; 23/06/2015
                              <1>       ; 18/04/2015
                              <1>       ; 12/04/2015
                              <1>       ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
                              <1>       ;
                              <1>       ; INPUT ->
                              <1>       ;      EAX = physical page address
                              <1>       ;            (use present value if EAX = 0)
                              <1>       ;      EBX = virtual (linear) address
                              <1>       ;      ECX = page attributes (lower 12 bits)
                              <1>       ;            (higher 20 bits must be ZERO)
                              <1>       ;            (bit 0 must be 1)
                              <1>       ;      u.pgdir = page directory (physical) address
                              <1>       ; OUTPUT ->
                              <1>       ;      EAX = physical page address
                              <1>       ;      (EDX = PTE value)
                              <1>         ;      EBX = virtual address
                              <1>         ;
                              <1>         ;      CF = 1 -> error
                              <1>         ;
                              <1>         ; Modified Registers -> EAX, EDX
                              <1>         ;
00004C41 50                   <1>         push   eax
00004C42 A1[B8030300]         <1>         mov    eax, [u.pgdir] ; 20/07/2015
00004C47 E837000000           <1>         call   get_pde
                              <1>                ; EDX = PDE address
                              <1>                ; EAX = PDE value
00004C4C 5A                   <1>         pop    edx ; physical page address
00004C4D 722A                 <1>         jc     short spte_err ; PDE not present
                              <1>         ;
00004C4F 53                   <1>         push   ebx ; 24/07/2015
00004C50 662500F0             <1>         and    ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
                              <1>                               ; EDX = PT address (physical)
00004C54 C1EB0C               <1>         shr    ebx, PAGE_SHIFT ; 12
00004C57 81E3FF030000         <1>         and    ebx, PTE_MASK; 03FFh
                              <1>                            ; clear higher 10 bits (PD bits)
00004C5D C1E302               <1>         shl    ebx, 2   ; offset to page table (1024*4)
00004C60 01C3                 <1>         add    ebx, eax
                              <1>         ;
00004C62 8B03                 <1>         mov    eax, [ebx] ; Old PTE value
00004C64 A801                 <1>         test   al, 1
00004C66 740C                 <1>         jz     short spte_0
00004C68 09D2                 <1>         or     edx, edx
00004C6A 750F                 <1>         jnz    short spte_1
00004C6C 662500F0             <1>         and    ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 bits
00004C70 89C2                 <1>         mov    edx, eax
00004C72 EB09                 <1>         jmp    short spte_2
                              <1> spte_0:
                              <1>         ; If this PTE contains a swap (disk) address,
                              <1>         ; it can be updated by using 'swap_in' procedure
                              <1>         ; only!
00004C74 21C0                 <1>         and    eax, eax
00004C76 7403                 <1>         jz     short spte_1
                              <1>         ; 24/07/2015
                              <1>         ; swapped page ! (on disk)
00004C78 5B                   <1>         pop    ebx
                              <1> spte_err:
00004C79 F9                   <1>         stc
00004C7A C3                   <1>         retn
                              <1> spte_1:
00004C7B 89D0                 <1>         mov    eax, edx
                              <1> spte_2:
00004C7D 09CA                 <1>         or     edx, ecx
                              <1>         ; 23/06/2015
00004C7F 8913                 <1>         mov    [ebx], edx ; PTE value in EDX
                              <1>         ; 24/07/2015
00004C81 5B                   <1>         pop    ebx
00004C82 C3                   <1>         retn
                              <1>
                              <1> get_pde:     ; Get present value of the relevant PDE
                              <1>         ; 20/07/2015
                              <1>         ; 18/04/2015
                              <1>         ; 12/04/2015
                              <1>         ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
                              <1>         ;
                              <1>         ; INPUT ->
                              <1>         ;      EBX = virtual (linear) address
                              <1>         ;      EAX = page directory (physical) address
                              <1>         ; OUTPUT ->
                              <1>         ;      EDX = Page directory entry address
                              <1>         ;      EAX = Page directory entry value
                              <1>         ;      CF = 1 -> PDE not present or invalid ?
                              <1>         ; Modified Registers -> EDX, EAX
                              <1>         ;
00004C83 89DA                 <1>         mov    edx, ebx
00004C85 C1EA16               <1>         shr    edx, PAGE_D_SHIFT ; 22  (12+10)
00004C88 C1E202               <1>         shl    edx, 2 ; offset to page directory (1024*4)
00004C8B 01C2                 <1>         add    edx, eax ; page directory address (physical)
00004C8D 8B02                 <1>         mov    eax, [edx]
00004C8F A801                 <1>         test   al, PDE_A_PRESENT ; page table is present or not !
00004C91 751F                 <1>         jnz    short gpte_retn
00004C93 F9                   <1>         stc
                              <1> gpde_retn:
00004C94 C3                   <1>         retn
                              <1>
                              <1> get_pte:
                              <1>                ; Get present value of the relevant PTE
                              <1>         ; 29/07/2015
                              <1>         ; 20/07/2015
                              <1>         ; 18/04/2015
                              <1>         ; 12/04/2015
                              <1>         ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
                              <1>         ;
                              <1>         ; INPUT ->
                              <1>         ;      EBX = virtual (linear) address
                              <1>         ;      EAX = page directory (physical) address
                              <1>         ; OUTPUT ->
                              <1>         ;      EDX = Page table entry address (if CF=0)
                              <1>         ;            Page directory entry address (if CF=1)
                              <1>         ;             (Bit 0 value is 0 if PT is not present)
                              <1>         ;      EAX = Page table entry value (page address)
                              <1>         ;      CF = 1 -> PDE not present or invalid ?
                              <1>         ; Modified Registers -> EAX, EDX
                              <1>         ;
00004C95 E8E9FFFFFF           <1>         call   get_pde
00004C9A 72F8                 <1>         jc     short gpde_retn     ; page table is not present
                              <1>         ;jnc   short gpte_1
                              <1>         ;retn
                              <1> ;gpte_1:
```

```
00004C9C 662500F0             <1>         and    ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
00004CA0 89DA                 <1>         mov    edx, ebx
00004CA2 C1EA0C               <1>         shr    edx, PAGE_SHIFT ; 12
00004CA5 81E2FF030000         <1>         and    edx, PTE_MASK; 03FFh
                              <1>                       ; clear higher 10 bits (PD bits)
00004CAB C1E202               <1>         shl    edx, 2 ; offset from start of page table (1024*4)
00004CAE 01C2                 <1>         add    edx, eax
00004CB0 8B02                 <1>         mov    eax, [edx]
                              <1> gpte_retn:
00004CB2 C3                   <1>         retn
                              <1> 
                              <1> deallocate_page_dir:
                              <1>         ; 15/09/2015
                              <1>         ; 05/08/2015
                              <1>         ; 30/04/2015
                              <1>         ; 28/04/2015
                              <1>         ; 17/10/2014
                              <1>         ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
                              <1>         ;
                              <1>         ; INPUT ->
                              <1>         ;       EAX = PHYSICAL ADDRESS OF THE PAGE DIRECTORY (CHILD)
                              <1>         ;       EBX = PHYSICAL ADDRESS OF THE PARENT'S PAGE DIRECTORY
                              <1>         ; OUTPUT ->
                              <1>         ;       All of page tables in the page directory
                              <1>         ;       and page dir's itself will be deallocated
                              <1>         ;       except 'read only' duplicated pages (will be converted
                              <1>         ;       to writable pages).
                              <1>         ;
                              <1>         ; Modified Registers -> EAX
                              <1>         ;
                              <1>         ;
00004CB3 56                   <1>         push   esi
00004CB4 51                   <1>         push   ecx
00004CB5 50                   <1>         push   eax
00004CB6 89C6                 <1>         mov    esi, eax
00004CB8 31C9                 <1>         xor    ecx, ecx
                              <1>         ; The 1st PDE points to Kernel Page Table 0 (the 1st 4MB),
                              <1>         ; it must not be deallocated
00004CBA 890E                 <1>         mov    [esi], ecx ; 0 ; clear PDE 0
                              <1> dapd_0:
00004CBC AD                   <1>         lodsd
00004CBD A801                 <1>         test   al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
00004CBF 7409                 <1>         jz     short dapd_1
00004CC1 662500F0             <1>         and    ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
00004CC5 E812000000           <1>         call   deallocate_page_table
                              <1> dapd_1:
00004CCA 41                   <1>         inc    ecx ; page directory entry index
00004CCB 81F900040000         <1>         cmp    ecx, PAGE_SIZE / 4 ; 1024
00004CD1 72E9                 <1>         jb     short dapd_0
                              <1> dapd_2:
00004CD3 58                   <1>         pop    eax
00004CD4 E87F000000           <1>         call   deallocate_page     ; deallocate the page dir's itself
00004CD9 59                   <1>         pop    ecx
00004CDA 5E                   <1>         pop    esi
00004CDB C3                   <1>         retn
                              <1> 
                              <1> deallocate_page_table:
                              <1>         ; 12/07/2016
                              <1>         ; 19/09/2015
                              <1>         ; 15/09/2015
                              <1>         ; 05/08/2015
                              <1>         ; 30/04/2015
                              <1>         ; 28/04/2015
                              <1>         ; 24/10/2014
                              <1>         ; 23/10/2014
                              <1>         ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
                              <1>         ;
                              <1>         ; INPUT ->
                              <1>         ;       EAX = PHYSICAL (real/flat) ADDRESS OF THE PAGE TABLE
                              <1>         ;       EBX = PHYSICAL ADDRESS OF THE PARENT'S PAGE DIRECTORY
                              <1>         ;       (ECX = page directory entry index)
                              <1>         ; OUTPUT ->
                              <1>         ;       All of pages in the page table and page table's itself
                              <1>         ;       will be deallocated except 'read only' duplicated pages
                              <1>         ;       (will be converted to writable pages).
                              <1>         ;
                              <1>         ; Modified Registers -> EAX
                              <1>         ;
00004CDC 56                   <1>         push   esi
00004CDD 57                   <1>         push   edi
00004CDE 52                   <1>         push   edx
00004CDF 50                   <1>         push   eax ; *
00004CE0 89C6                 <1>         mov    esi, eax
00004CE2 31FF                 <1>         xor    edi, edi ; 0
                              <1> dapt_0:
00004CE4 AD                   <1>         lodsd
00004CE5 A801                 <1>         test   al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
00004CE7 7441                 <1>         jz     short dapt_1
                              <1>         ;
00004CE9 A802                 <1>         test   al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
                              <1>                               ; (must be 1)
00004CEB 754C                 <1>         jnz    short dapt_3
                              <1>         ; Read only -duplicated- page (belongs to a parent or a child)
00004CED 66A90002             <1>           test    ax, PTE_DUPLICATED ; Was this page duplicated
                              <1>                                ; as child's page ?
00004CF1 7451                 <1>         jz     short dapt_4 ; Clear PTE but don't deallocate the page!
                              <1>         ; check the parent's PTE value is read only & same page or not..
                              <1>         ; ECX = page directory entry index (0-1023)
00004CF3 53                   <1>         push   ebx
00004CF4 51                   <1>         push   ecx
00004CF5 66C1E102             <1>         shl    cx, 2 ; *4
00004CF9 01CB                 <1>         add    ebx, ecx ; PDE offset (for the parent)
00004CFB 8B0B                 <1>         mov    ecx, [ebx]
```

```
00004CFD F6C101                  <1>       test   cl, PDE_A_PRESENT ; present (valid) or not ?
00004D00 7435                    <1>       jz     short dapt_2 ; parent process does not use this page
00004D02 6681E100F0              <1>       and    cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
                                 <1>       ; EDI = page table entry index (0-1023)
00004D07 89FA                    <1>       mov    edx, edi
00004D09 66C1E202                <1>       shl    dx, 2 ; *4
00004D0D 01CA                    <1>       add    edx, ecx ; PTE offset (for the parent)
00004D0F 8B1A                    <1>       mov    ebx, [edx]
00004D11 F6C301                  <1>       test   bl, PTE_A_PRESENT ; present or not ?
00004D14 7421                    <1>       jz     short dapt_2 ; parent process does not use this page
00004D16 662500F0                <1>       and    ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
00004D1A 6681E300F0              <1>       and    bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
00004D1F 39D8                    <1>       cmp    eax, ebx     ; parent's and child's pages are same ?
00004D21 7514                    <1>       jne    short dapt_2 ; not same page
                                 <1>                           ; deallocate the child's page
00004D23 800A02                  <1>         or      byte [edx], PTE_A_WRITE ; convert to writable page (parent)
00004D26 59                      <1>       pop    ecx
00004D27 5B                      <1>       pop    ebx
00004D28 EB1A                    <1>       jmp    short dapt_4
                                 <1> dapt_1:
00004D2A 09C0                    <1>       or     eax, eax     ; swapped page ?
00004D2C 741D                    <1>       jz     short dapt_5 ; no
                                 <1>                           ; yes
00004D2E D1E8                    <1>       shr    eax, 1
00004D30 E8CA040000              <1>       call   unlink_swap_block ; Deallocate swapped page block
                                 <1>                             ; on the swap disk (or in file)
00004D35 EB14                    <1>       jmp    short dapt_5
                                 <1> dapt_2:
00004D37 59                      <1>       pop    ecx
00004D38 5B                      <1>       pop    ebx
                                 <1> dapt_3:
                                 <1>       ; 12/07/2016
00004D39 66A90004                <1>       test   ax, PTE_SHARED ; shared or direct memory access indicator
00004D3D 7505                    <1>       jnz    short dapt_4   ; AVL bit 1 = 1, do not deallocate this page!
                                 <1>       ;
                                 <1>       ;and   ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
00004D3F E814000000              <1>       call   deallocate_page ; set the mem allocation bit of this page
                                 <1> dapt_4:
00004D44 C746FC00000000          <1>       mov    dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)
                                 <1> dapt_5:
00004D4B 47                      <1>       inc    edi ; page table entry index
00004D4C 81FF00040000            <1>       cmp    edi, PAGE_SIZE / 4 ; 1024
00004D52 7290                    <1>       jb     short dapt_0
                                 <1>       ;
00004D54 58                      <1>       pop    eax ; *
00004D55 5A                      <1>       pop    edx
00004D56 5F                      <1>       pop    edi
00004D57 5E                      <1>       pop    esi
                                 <1>       ;
                                 <1>       ;call  deallocate_page     ; deallocate the page table's itself
                                 <1>       ;retn
                                 <1>
                                 <1> deallocate_page:
                                 <1>       ; 15/09/2015
                                 <1>       ; 28/04/2015
                                 <1>       ; 10/03/2015
                                 <1>       ; 17/10/2014
                                 <1>       ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
                                 <1>       ;
                                 <1>       ; INPUT ->
                                 <1>       ;      EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
                                 <1>       ; OUTPUT ->
                                 <1>       ;      [free_pages] is increased
                                 <1>       ;      (corresponding MEMORY ALLOCATION TABLE bit is SET)
                                 <1>       ;      CF = 1 if the page is already deallocated
                                 <1>       ;             (or not allocated) before.
                                 <1>       ;
                                 <1>       ; Modified Registers -> EAX
                                 <1>       ;
00004D58 53                      <1>       push   ebx
00004D59 52                      <1>       push   edx
                                 <1>       ;
00004D5A C1E80C                  <1>       shr    eax, PAGE_SHIFT      ; shift physical address to
                                 <1>                                ; 12 bits right
                                 <1>                                ; to get page number
00004D5D 89C2                    <1>       mov    edx, eax
                                 <1>       ; 15/09/2015
00004D5F C1EA03                  <1>       shr    edx, 3            ; to get offset to M.A.T.
                                 <1>                                ; (1 allocation bit = 1 page)
                                 <1>                                ; (1 allocation bytes = 8 pages)
00004D62 80E2FC                  <1>       and    dl, 0FCh          ; clear lower 2 bits
                                 <1>                                ; (to get 32 bit position)
                                 <1>       ;
00004D65 BB00001000              <1>       mov    ebx, MEM_ALLOC_TBL   ; Memory Allocation Table address
00004D6A 01D3                    <1>       add    ebx, edx
00004D6C 83E01F                  <1>       and    eax, 1Fh          ; lower 5 bits only
                                 <1>                                ; (allocation bit position)
00004D6F 3B15[44580100]          <1>       cmp    edx, [next_page]     ; is the new free page address lower
                                 <1>                                ; than the address in 'next_page' ?
                                 <1>                                ; (next/first free page value)
00004D75 7306                    <1>       jnb    short dap_1       ; no
00004D77 8915[44580100]          <1>       mov    [next_page], edx     ; yes
                                 <1> dap_1:
00004D7D 0FAB03                  <1>       bts    [ebx], eax        ; unlink/release/deallocate page
                                 <1>                                ; set relevant bit to 1.
                                 <1>                                ; set CF to the previous bit value
                                 <1>       ;cmc                     ; complement carry flag
                                 <1>       ;jc    short dap_2       ; do not increase free_pages count
                                 <1>                                ; if the page is already deallocated
                                 <1>                                ; before.
00004D80 FF05[40580100]          <1>         inc     dword [free_pages]
                                 <1> dap_2:
00004D86 5A                      <1>       pop    edx
```

```
00004D87 5B                  <1>       pop    ebx
00004D88 C3                  <1>       retn
                             <1> 
                             <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             <1> ;;                                                              ;;
                             <1> ;; Copyright (C) KolibriOS team 2004-2012. All rights reserved. ;;
                             <1> ;; Distributed under terms of the GNU General Public License   ;;
                             <1> ;;                                                              ;;
                             <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             <1> 
                             <1> ;;$Revision: 5057 $
                             <1> 
                             <1> 
                             <1> ;;align 4
                             <1> ;;proc alloc_page
                             <1> 
                             <1> ;;        pushfd
                             <1> ;;        cli
                             <1> ;;        push    ebx
                             <1> ;;;//-
                             <1> ;;        cmp     [pg_data.pages_free], 1
                             <1> ;;        jle     .out_of_memory
                             <1> ;;;//-
                             <1> ;;
                             <1> ;;        mov     ebx, [page_start]
                             <1> ;;        mov     ecx, [page_end]
                             <1> ;;.l1:
                             <1> ;;        bsf     eax, [ebx];
                             <1> ;;        jnz     .found
                             <1> ;;        add     ebx, 4
                             <1> ;;        cmp     ebx, ecx
                             <1> ;;        jb      .l1
                             <1> ;;        pop     ebx
                             <1> ;;        popfd
                             <1> ;;        xor     eax, eax
                             <1> ;;        ret
                             <1> ;;.found:
                             <1> ;;;//-
                             <1> ;;        dec     [pg_data.pages_free]
                             <1> ;;        jz      .out_of_memory
                             <1> ;;;//-
                             <1> ;;        btr     [ebx], eax
                             <1> ;;        mov     [page_start], ebx
                             <1> ;;        sub     ebx, sys_pgmap
                             <1> ;;        lea     eax, [eax+ebx*8]
                             <1> ;;        shl     eax, 12
                             <1> ;;;//-       dec [pg_data.pages_free]
                             <1> ;;        pop     ebx
                             <1> ;;        popfd
                             <1> ;;        ret
                             <1> ;;;//-
                             <1> ;;.out_of_memory:
                             <1> ;;        mov     [pg_data.pages_free], 1
                             <1> ;;        xor     eax, eax
                             <1> ;;        pop     ebx
                             <1> ;;        popfd
                             <1> ;;        ret
                             <1> ;;;//-
                             <1> ;;endp
                             <1> 
                             <1> duplicate_page_dir:
                             <1>       ; 21/09/2015
                             <1>       ; 31/08/2015
                             <1>       ; 20/07/2015
                             <1>       ; 28/04/2015
                             <1>       ; 27/04/2015
                             <1>       ; 18/04/2015
                             <1>       ; 12/04/2015
                             <1>       ; 18/10/2014
                             <1>       ; 16/10/2014 (Retro UNIX 386 v1 - beginning)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;     [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's
                             <1>       ;                page directory.
                             <1>       ; OUTPUT ->
                             <1>       ;     EAX =  PHYSICAL (real/flat) ADDRESS of the child's
                             <1>       ;            page directory.
                             <1>       ;     (New page directory with new page table entries.)
                             <1>       ;     (New page tables with read only copies of the parent's
                             <1>       ;     pages.)
                             <1>       ;     EAX = 0 -> Error (CF = 1)
                             <1>       ;
                             <1>       ; Modified Registers -> none (except EAX)
                             <1>       ;
00004D89 E8ECFDFFFF          <1>       call   allocate_page
00004D8E 723E                <1>       jc     short dpd_err
                             <1>       ;
00004D90 55                  <1>       push   ebp ; 20/07/2015
00004D91 56                  <1>       push   esi
00004D92 57                  <1>       push   edi
00004D93 53                  <1>       push   ebx
00004D94 51                  <1>       push   ecx
00004D95 8B35[B8030300]      <1>       mov    esi, [u.pgdir]
00004D9B 89C7                <1>       mov    edi, eax
00004D9D 50                  <1>       push   eax ; save child's page directory address
                             <1>       ; 31/08/2015
                             <1>       ; copy PDE 0 from the parent's page dir to the child's page dir
                             <1>       ; (use same system space for all user page tables)
00004D9E A5                  <1>       movsd
00004D9F BD00004000          <1>       mov    ebp, 1024*4096 ; pass the 1st 4MB (system space)
00004DA4 B9FF030000          <1>       mov    ecx, (PAGE_SIZE / 4) - 1 ; 1023
                             <1> dpd_0:
00004DA9 AD                  <1>       lodsd
```

```
                             <1>       ;or    eax, eax
                             <1>         ;jnz     short dpd_1
00004DAA A801                <1>       test   al, PDE_A_PRESENT ;  bit 0 =  1
00004DAC 7508                <1>       jnz    short dpd_1
                             <1>       ; 20/07/2015 (virtual address at the end of the page table)
00004DAE 81C500004000        <1>       add    ebp, 1024*4096 ; page size * PTE count
00004DB4 EB0F                <1>       jmp    short dpd_2
                             <1> dpd_1:
00004DB6 662500F0            <1>       and    ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
00004DBA 89C3                <1>       mov    ebx, eax
                             <1>       ; EBX = Parent's page table address
00004DBC E81F000000          <1>       call   duplicate_page_table
00004DC1 720C                <1>       jc     short dpd_p_err
                             <1>       ; EAX = Child's page table address
00004DC3 0C07                <1>       or     al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
                             <1>                      ; set bit 0, bit 1 and bit 2 to 1
                             <1>                      ; (present, writable, user)
                             <1> dpd_2:
00004DC5 AB                  <1>       stosd
00004DC6 E2E1                <1>       loop   dpd_0
                             <1>       ;
00004DC8 58                  <1>       pop    eax  ; restore child's page directory address
                             <1> dpd_3:
00004DC9 59                  <1>       pop    ecx
00004DCA 5B                  <1>       pop    ebx
00004DCB 5F                  <1>       pop    edi
00004DCC 5E                  <1>       pop    esi
00004DCD 5D                  <1>       pop    ebp ; 20/07/2015
                             <1> dpd_err:
00004DCE C3                  <1>       retn
                             <1> dpd_p_err:
                             <1>       ; release the allocated pages missing (recover free space)
00004DCF 58                  <1>       pop    eax  ; the new page directory address (physical)
00004DD0 8B1D[B8030300]      <1>       mov    ebx, [u.pgdir] ; parent's page directory address
00004DD6 E8D8FEFFFF          <1>       call   deallocate_page_dir
00004DDB 29C0                <1>       sub    eax, eax ; 0
00004DDD F9                  <1>       stc
00004DDE EBE9                <1>       jmp    short dpd_3
                             <1>
                             <1> duplicate_page_table:
                             <1>       ; 20/02/2017
                             <1>       ; 21/09/2015
                             <1>       ; 20/07/2015
                             <1>       ; 05/05/2015
                             <1>       ; 28/04/2015
                             <1>       ; 27/04/2015
                             <1>       ; 18/04/2015
                             <1>       ; 18/10/2014
                             <1>       ; 16/10/2014 (Retro UNIX 386 v1 - beginning)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.
                             <1>       ;        20/02/2017
                             <1>       ;      EBP = Linear address of the page (from 'duplicate_page_dir')
                             <1>       ;            (Linear address = CORE + user's virtual address)
                             <1>       ; OUTPUT ->
                             <1>       ;      EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.
                             <1>       ;            (with 'read only' attribute of page table entries)
                             <1>       ;      20/02/2017
                             <1>       ;      EBP = Next linear page address (for 'duplicate_page_dir')
                             <1>       ;
                             <1>       ;      CF = 1 -> error
                             <1>       ;
                             <1>       ; Modified Registers -> EBP (except EAX)
                             <1>       ;
00004DE0 E895FDFFFF          <1>       call   allocate_page
00004DE5 726A                <1>       jc     short dpt_err
                             <1>       ;
00004DE7 50                  <1>       push   eax ; *
00004DE8 56                  <1>       push   esi
00004DE9 57                  <1>       push   edi
00004DEA 52                  <1>       push   edx
00004DEB 51                  <1>       push   ecx
                             <1>       ;
00004DEC 89DE                <1>       mov    esi, ebx
00004DEE 89C7                <1>       mov    edi, eax
00004DF0 89C2                <1>       mov    edx, eax
00004DF2 81C200100000        <1>       add    edx, PAGE_SIZE
                             <1> dpt_0:
00004DF8 AD                  <1>       lodsd
00004DF9 21C0                <1>       and    eax, eax
00004DFB 7444                <1>       jz     short dpt_3
00004DFD A801                <1>       test   al, PTE_A_PRESENT ;  bit 0 =  1
00004DFF 7507                <1>       jnz    short dpt_1
                             <1>       ; 20/07/2015
                             <1>       ; ebp = virtual (linear) address of the memory page
00004E01 E83F050000          <1>       call   reload_page ; 28/04/2015
00004E06 7244                <1>       jc     short dpt_p_err
                             <1> dpt_1:
                             <1>       ; 21/09/2015
00004E08 89C1                <1>       mov    ecx, eax
00004E0A 662500F0            <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
00004E0E F6C102              <1>       test   cl, PTE_A_WRITE ; writable page ?
00004E11 7525                <1>       jnz    short dpt_2
                             <1>       ; Read only (parent) page
                             <1>       ;      - there is a third process which uses this page -
                             <1>       ; Allocate a new page for the child process
00004E13 E862FDFFFF          <1>       call   allocate_page
00004E18 7232                <1>       jc     short dpt_p_err
00004E1A 57                  <1>       push   edi
00004E1B 56                  <1>       push   esi
00004E1C 89CE                <1>       mov    esi, ecx
00004E1E 89C7                <1>       mov    edi, eax
```

```
00004E20 B900040000               <1>       mov    ecx, PAGE_SIZE/4
00004E25 F3A5                     <1>       rep    movsd  ; copy page (4096 bytes)
00004E27 5E                       <1>       pop    esi
00004E28 5F                       <1>       pop    edi
                                  <1>       ;
00004E29 53                       <1>       push   ebx
00004E2A 50                       <1>       push   eax
                                  <1>       ; 20/07/2015
00004E2B 89EB                     <1>       mov    ebx, ebp
                                  <1>       ; ebx = virtual (linear) address of the memory page
00004E2D E887030000               <1>       call   add_to_swap_queue
00004E32 58                       <1>       pop    eax
00004E33 5B                       <1>       pop    ebx
                                  <1>       ; 21/09/2015
00004E34 0C07                     <1>       or     al, PTE_A_USER+PTE_A_WRITE+PTE_A_PRESENT
                                  <1>              ; user + writable + present page
00004E36 EB09                     <1>       jmp    short dpt_3
                                  <1> dpt_2:
                                  <1>       ;or    ax, PTE_A_USER+PTE_A_PRESENT
00004E38 0C05                     <1>       or     al, PTE_A_USER+PTE_A_PRESENT
                                  <1>                  ; (read only page!)
00004E3A 8946FC                   <1>       mov    [esi-4], eax ; update parent's PTE
00004E3D 660D0002                 <1>       or      ax, PTE_DUPLICATED  ; (read only page & duplicated PTE!)
                                  <1> dpt_3:
00004E41 AB                       <1>       stosd  ; EDI points to child's PTE
                                  <1>       ;
00004E42 81C500100000             <1>       add    ebp, 4096 ; 20/07/2015 (next page)
                                  <1>       ;
00004E48 39D7                     <1>       cmp    edi, edx
00004E4A 72AC                     <1>       jb     short dpt_0
                                  <1> dpt_p_err:
00004E4C 59                       <1>       pop    ecx
00004E4D 5A                       <1>       pop    edx
00004E4E 5F                       <1>       pop    edi
00004E4F 5E                       <1>       pop    esi
00004E50 58                       <1>       pop    eax ; *
                                  <1> dpt_err:
00004E51 C3                       <1>       retn
                                  <1>
                                  <1> page_fault_handler:       ; CPU EXCEPTION 0Eh (14) : Page Fault !
                                  <1>       ; 21/09/2015
                                  <1>       ; 19/09/2015
                                  <1>       ; 17/09/2015
                                  <1>       ; 28/08/2015
                                  <1>       ; 20/07/2015
                                  <1>       ; 28/06/2015
                                  <1>       ; 03/05/2015
                                  <1>       ; 30/04/2015
                                  <1>       ; 18/04/2015
                                  <1>       ; 12/04/2015
                                  <1>       ; 30/10/2014
                                  <1>       ; 11/09/2014
                                  <1>       ; 10/09/2014 (Retro UNIX 386 v1 - beginning)
                                  <1>       ;
                                  <1>       ; Note: This is not an interrupt/exception handler.
                                  <1>       ;      This is a 'page fault remedy' subroutine
                                  <1>       ;      which will be called by standard/uniform
                                  <1>       ;      exception handler.
                                  <1>       ;
                                  <1>       ; INPUT ->
                                  <1>       ;      [error_code] = 32 bit ERROR CODE (lower 5 bits are valid)
                                  <1>       ;
                                  <1>       ;      cr2 = the virtual (linear) address
                                  <1>       ;            which has caused to page fault (19/09/2015)
                                  <1>       ;
                                  <1>       ; OUTPUT ->
                                  <1>       ;      (corresponding PAGE TABLE ENTRY is mapped/set)
                                  <1>       ;      EAX = 0 -> no error
                                  <1>       ;      EAX > 0 -> error code in EAX (also CF = 1)
                                  <1>       ;
                                  <1>       ; Modified Registers -> none (except EAX)
                                  <1>       ;
                                  <1>         ;
                                  <1>         ; ERROR CODE:
                                  <1>       ;       31  .....   4   3  2   1  0
                                  <1>       ;      +---+-- --+---+---+---+---+---+---+
                                  <1>       ;      |   Reserved  | I | R | U | W | P |
                                  <1>       ;      +---+-- --+---+---+---+---+---+---+
                                  <1>       ;
                                  <1>       ; P : PRESENT -     When set, the page fault was caused by
                                  <1>       ;            a page-protection violation. When not set,
                                  <1>       ;            it was caused by a non-present page.
                                  <1>       ; W : WRITE   -     When set, the page fault was caused by
                                  <1>       ;            a page write. When not set, it was caused
                                  <1>       ;            by a page read.
                                  <1>       ; U : USER    -     When set, the page fault was caused
                                  <1>       ;            while CPL = 3.
                                  <1>       ;            This does not necessarily mean that
                                  <1>       ;            the page fault was a privilege violation.
                                  <1>       ; R : RESERVD -     When set, the page fault was caused by
                                  <1>       ;     WRITE  reading a 1 in a reserved field.
                                  <1>       ; I : INSTRUC -     When set, the page fault was caused by
                                  <1>       ;     FETCH  an instruction fetch
                                  <1>       ;
                                  <1>       ;; x86 (32 bit) VIRTUAL ADDRESS TRANSLATION
                                  <1>       ;  31               22                  12 11                    0
                                  <1>       ; +-------------------+-------------------+-----------------------+
                                  <1>             ; | PAGE DIR. ENTRY # | PAGE TAB. ENTRY # |        OFFSET         |
                                  <1>             ; +-------------------+-------------------+-----------------------+
                                  <1>       ;
                                  <1>
                                  <1>       ;; CR3 REGISTER (Control Register 3)
                                  <1>       ;  31                                   12             5 4 3 2   0
```

```
                             <1>       ; +--------------------------------------+-------------+---+-----+
                             <1>             ; |                                       |          |P|P|     |
                             <1>             ; |   PAGE DIRECTORY TABLE BASE ADDRESS   |  reserved     |C|W|rsvrd|
                             <1>             ; |                                       |          |D|T|     |
                             <1>       ; +--------------------------------------+-------------+---+-----+
                             <1>       ;
                             <1>       ;     PWT    - WRITE THROUGH
                             <1>       ;     PCD    - CACHE DISABLE
                             <1>       ;
                             <1>       ;
                             <1>       ;; x86 PAGE DIRECTORY ENTRY (4 KByte Page)
                             <1>       ;  31                                   12 11  9 8 7 6 5 4 3 2 1 0
                             <1>       ; +--------------------------------------+-----+---+-+-+---+-+-+-+
                             <1>             ; |                                       |     | | | | |P|P|U|R| |
                             <1>             ; |     PAGE TABLE BASE ADDRESS 31..12    | AVL |G|0|D|A|C|W|/|/|P|
                             <1>             ; |                                       |     | | | | |D|T|S|W| |
                             <1>       ; +--------------------------------------+-----+---+-+-+---+-+-+-+
                             <1>       ;
                             <1>         ;       P      - PRESENT
                             <1>         ;       R/W    - READ/WRITE
                             <1>         ;       U/S    - USER/SUPERVISOR
                             <1>       ;     PWT    - WRITE THROUGH
                             <1>       ;     PCD    - CACHE DISABLE
                             <1>       ;     A      - ACCESSED
                             <1>         ;       D      - DIRTY (IGNORED)
                             <1>       ;     PAT    - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
                             <1>       ;     G      - GLOBAL     (IGNORED)
                             <1>         ;       AVL    - AVAILABLE FOR SYSTEMS PROGRAMMER USE
                             <1>       ;
                             <1>       ;
                             <1>       ;; x86 PAGE TABLE ENTRY (4 KByte Page)
                             <1>       ;  31                                   12 11  9 8 7 6 5 4 3 2 1 0
                             <1>       ; +--------------------------------------+-----+---+-+-+---+-+-+-+
                             <1>             ; |                                       |     | |P| | |P|P|U|R| |
                             <1>             ; |     PAGE FRAME BASE ADDRESS 31..12    | AVL |G|A|D|A|C|W|/|/|P|
                             <1>             ; |                                       |     | |T| | |D|T|S|W| |
                             <1>       ; +--------------------------------------+-----+---+-+-+---+-+-+-+
                             <1>       ;
                             <1>         ;       P      - PRESENT
                             <1>         ;       R/W    - READ/WRITE
                             <1>         ;       U/S    - USER/SUPERVISOR
                             <1>       ;     PWT    - WRITE THROUGH
                             <1>       ;     PCD    - CACHE DISABLE
                             <1>       ;     A      - ACCESSED
                             <1>         ;       D      - DIRTY
                             <1>       ;     PAT    - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
                             <1>       ;     G      - GLOBAL
                             <1>         ;       AVL    - AVAILABLE FOR SYSTEMS PROGRAMMER USE
                             <1>       ;
                             <1>       ;
                             <1>       ;; 80386 PAGE TABLE ENTRY (4 KByte Page)
                             <1>       ;  31                                   12 11  9 8 7 6 5 4 3 2 1 0
                             <1>       ; +--------------------------------------+-----+-+-+-+-+---+-+-+-+
                             <1>             ; |                                       |     | | | | | | |U|R| |
                             <1>             ; |     PAGE FRAME BASE ADDRESS 31..12    | AVL |0|0|D|A|0|0|/|/|P|
                             <1>             ; |                                       |     | | | | | | |S|W| |
                             <1>             ; +--------------------------------------+-----+-+-+-+-+---+-+-+-+
                             <1>       ;
                             <1>         ;       P      - PRESENT
                             <1>         ;       R/W    - READ/WRITE
                             <1>         ;       U/S    - USER/SUPERVISOR
                             <1>         ;       D      - DIRTY
                             <1>         ;       AVL    - AVAILABLE FOR SYSTEMS PROGRAMMER USE
                             <1>       ;
                             <1>         ;       NOTE: 0 INDICATES INTEL RESERVED. DO NOT DEFINE.
                             <1>       ;
                             <1>       ;
                             <1>       ;; Invalid Page Table Entry
                             <1>       ; 31                                                         1 0
                             <1>             ; +-------------------------------------------------------------+-+
                             <1>             ; |                                                             | |
                             <1>             ; |                          AVAILABLE                          |0|
                             <1>             ; |                                                             | |
                             <1>             ; +-------------------------------------------------------------+-+
                             <1>       ;
                             <1>
00004E52 53                  <1>       push   ebx
00004E53 52                  <1>       push   edx
00004E54 51                  <1>       push   ecx
                             <1>       ;
                             <1>       ; 21/09/2015 (debugging)
00004E55 FF05[CC030300]      <1>       inc    dword [u.pfcount] ; page fault count for running process
00004E5B FF05[80050300]      <1>       inc    dword [PF_Count] ; total page fault count
                             <1>       ; 28/06/2015
                             <1>       ;mov   edx, [error_code] ; Lower 5 bits are valid
00004E61 8A15[78050300]      <1>       mov    dl, [error_code]
                             <1>       ;
00004E67 F6C201              <1>       test   dl, 1  ; page fault was caused by a non-present page
                             <1>                     ; sign
00004E6A 7422                <1>       jz     short pfh_alloc_np
                             <1>       ;
                             <1>       ; If it is not a 'write on read only page' type page fault
                             <1>       ; major page fault error with minor reason must be returned without
                             <1>       ; fixing the problem. 'sys_exit with error' will be needed
                             <1>       ; after return here!
                             <1>       ; Page fault will be remedied, by copying page contents
                             <1>       ; to newly allocated page with write permission;
                             <1>       ; sys_fork -> sys_exec -> copy on write, demand paging method is
                             <1>       ; used for working with minimum possible memory usage.
                             <1>       ; sys_fork will duplicate page directory and tables of parent
                             <1>       ; process with 'read only' flag. If the child process attempts to
                             <1>       ; write on these read only pages, page fault will be directed here
                             <1>       ; for allocating a new page with same data/content.
                             <1>       ;
                             <1>       ; IMPORTANT : Retro UNIX 386 v1 (and SINGLIX and TR-DOS)
                             <1>       ; will not force to separate CODE and DATA space
                             <1>       ; in a process/program...
                             <1>       ; CODE segment/section may contain DATA!
                             <1>       ; It is flat, smoth and simplest programming method already as in
                             <1>       ; Retro UNIX 8086 v1 and MS-DOS programs.
                             <1>       ;
00004E6C F6C202              <1>       test   dl, 2  ; page fault was caused by a page write
                             <1>                    ; sign
00004E6F 0F84AB000000        <1>         jz      pfh_p_err
                             <1>       ; 31/08/2015
00004E75 F6C204              <1>       test   dl, 4  ; page fault was caused while CPL = 3 (user mode)
                             <1>                    ; sign.  (U+W+P = 4+2+1 = 7)
00004E78 0F84A2000000        <1>         jz   pfh_pv_err
                             <1>       ;
                             <1>       ; make a new page and copy the parent's page content
                             <1>       ; as the child's new page content
                             <1>       ;
00004E7E 0F20D3              <1>       mov    ebx, cr2 ; CR2 contains the linear address
                             <1>                     ; which has caused to page fault
00004E81 E8A2000000          <1>       call   copy_page
00004E86 0F828D000000        <1>         jc      pfh_im_err ; insufficient memory
                             <1>       ;
00004E8C EB7D                <1>         jmp     pfh_cpp_ok
                             <1>       ;
                             <1> pfh_alloc_np:
00004E8E E8E7FCFFFF          <1>       call   allocate_page; (allocate a new page)
00004E93 0F8280000000        <1>         jc      pfh_im_err      ; 'insufficient memory' error
                             <1> pfh_chk_cpl:
                             <1>       ; EAX = Physical (base) address of the allocated (new) page
                             <1>              ; (Lower 12 bits are ZERO, because
                             <1>              ;      the address is on a page boundary)
00004E99 80E204              <1>       and    dl, 4  ; CPL = 3 ?
00004E9C 7505                <1>       jnz    short pfh_um
                             <1>                    ; Page fault handler for kernel/system mode (CPL=0)
00004E9E 0F20DB              <1>       mov    ebx, cr3 ; CR3 (Control Register 3) contains physical address
                             <1>                     ; of the current/active page directory
                             <1>                     ; (Always kernel/system mode page directory, here!)
                             <1>                     ; Note: Lower 12 bits are 0. (page boundary)
00004EA1 EB06                <1>       jmp    short pfh_get_pde
                             <1>       ;
                             <1> pfh_um:                   ; Page fault handler for user/appl. mode (CPL=3)
00004EA3 8B1D[B8030300]      <1>       mov    ebx, [u.pgdir] ; Page directory of current/active process
                             <1>                    ; Physical address of the USER's page directory
                             <1>                    ; Note: Lower 12 bits are 0. (page boundary)
                             <1> pfh_get_pde:
00004EA9 80CA03              <1>       or     dl, 3  ; USER + WRITE + PRESENT or SYSTEM + WRITE + PRESENT
00004EAC 0F20D1              <1>       mov    ecx, cr2 ; CR2 contains the virtual address
                             <1>                     ; which has been caused to page fault
                             <1>                     ;
00004EAF C1E914              <1>       shr    ecx, 20       ; shift 20 bits right
00004EB2 80E1FC              <1>       and    cl, 0FCh ; mask lower 2 bits to get PDE offset
                             <1>       ;
00004EB5 01CB                <1>       add    ebx, ecx ; now, EBX points to the relevant page dir entry
00004EB7 8B0B                <1>       mov    ecx, [ebx] ; physical (base) address of the page table
00004EB9 F6C101              <1>       test   cl, 1   ; check bit 0 is set (1) or not (0).
00004EBC 740B                <1>       jz     short pfh_set_pde ; Page directory entry is not valid,
                             <1>                             ; set/validate page directory entry
00004EBE 6681E100F0          <1>       and    cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
00004EC3 89CB                <1>       mov    ebx, ecx ; Physical address of the page table
00004EC5 89C1                <1>       mov    ecx, eax ; new page address (physical)
00004EC7 EB16                <1>       jmp    short pfh_get_pte
                             <1> pfh_set_pde:
                             <1>       ;; NOTE: Page directories and page tables never be swapped out!
                             <1>       ;;      (So, we know this PDE is empty or invalid)
                             <1>       ;
00004EC9 08D0                <1>       or     al, dl  ; lower 3 bits are used as U/S, R/W, P flags
00004ECB 8903                <1>       mov    [ebx], eax ; Let's put the new page directory entry here !
00004ECD 30C0                <1>       xor    al, al  ; clear lower (3..8) bits
00004ECF 89C3                <1>       mov    ebx, eax
00004ED1 E8A4FCFFFF          <1>       call   allocate_page ; (allocate a new page)
00004ED6 7241                <1>       jc     short pfh_im_err   ; 'insufficient memory' error
                             <1> pfh_spde_1:
                             <1>       ; EAX = Physical (base) address of the allocated (new) page
00004ED8 89C1                <1>       mov    ecx, eax
00004EDA E815FDFFFF          <1>       call   clear_page ; Clear page content
                             <1> pfh_get_pte:
00004EDF 0F20D0              <1>       mov    eax, cr2 ; virtual address
                             <1>                     ; which has been caused to page fault
00004EE2 89C7                <1>       mov    edi, eax ; 20/07/2015
00004EE4 C1E80C              <1>       shr    eax, 12       ; shift 12 bit right to get
                             <1>                     ; higher 20 bits of the page fault address
00004EE7 25FF030000          <1>       and    eax, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)
00004EEC C1E002              <1>       shl    eax, 2 ; shift 2 bits left to get PTE offset
00004EEF 01C3                <1>       add    ebx, eax ; now, EBX points to the relevant page table entry
00004EF1 8B03                <1>       mov    eax, [ebx] ; get previous value of pte
                             <1>              ; bit 0 of EAX is always 0 (otherwise we would not be here)
00004EF3 21C0                <1>       and    eax, eax
00004EF5 7410                <1>       jz     short pfh_gpte_1
                             <1>       ; 20/07/2015
00004EF7 87D9                <1>       xchg   ebx, ecx ; new page address (physical)
00004EF9 55                  <1>       push   ebp ; 20/07/2015
00004EFA 0F20D5              <1>       mov    ebp, cr2
                             <1>              ; ECX = physical address of the page table entry
                             <1>              ; EBX = Memory page address (physical!)
                             <1>              ; EAX = Swap disk (offset) address
                             <1>              ; EBP = virtual address (page fault address)
00004EFD E8B7000000          <1>       call   swap_in
00004F02 5D                  <1>       pop    ebp
00004F03 7210                <1>       jc      short pfh_err_retn
00004F05 87CB                <1>       xchg   ecx, ebx
                             <1>              ; EBX = physical address of the page table entry
                             <1>             ; ECX = new page
                             <1> pfh_gpte_1:
00004F07 08D1                <1>       or     cl, dl ; lower 3 bits are used as U/S, R/W, P flags
00004F09 890B                <1>       mov    [ebx], ecx ; Let's put the new page table entry here !
                             <1> pfh_cpp_ok:
                             <1>       ; 20/07/2015
00004F0B 0F20D3              <1>       mov    ebx, cr2
00004F0E E8A6020000          <1>       call   add_to_swap_queue
                             <1>       ;
                             <1>       ; The new PTE (which contains the new page) will be added to
                             <1>       ; the swap queue, here.
                             <1>       ; (Later, if memory will become insufficient,
                             <1>       ; one page will be swapped out which is at the head of
                             <1>       ; the swap queue by using FIFO and access check methods.)
                             <1>       ;
00004F13 31C0                <1>       xor    eax, eax  ; 0
                             <1>       ;
                             <1> pfh_err_retn:
00004F15 59                  <1>       pop    ecx
00004F16 5A                  <1>       pop    edx
00004F17 5B                  <1>       pop    ebx
00004F18 C3                  <1>       retn
                             <1>
                             <1> pfh_im_err:
00004F19 B8E4000000          <1>       mov    eax, ERR_MAJOR_PF + ERR_MINOR_IM ; Error code in AX
                             <1>                    ; Major (Primary) Error: Page Fault
                             <1>                    ; Minor (Secondary) Error: Insufficient Memory !
00004F1E EBF5                <1>       jmp    short pfh_err_retn
                             <1>
                             <1>
                             <1> pfh_p_err: ; 09/03/2015
                             <1> pfh_pv_err:
                             <1>       ; Page fault was caused by a protection-violation
00004F20 B8E6000000          <1>       mov    eax, ERR_MAJOR_PF + ERR_MINOR_PV ; Error code in AX
                             <1>                    ; Major (Primary) Error: Page Fault
                             <1>                    ; Minor (Secondary) Error: Protection violation !
00004F25 F9                  <1>       stc
00004F26 EBED                <1>       jmp    short pfh_err_retn
                             <1>
                             <1> copy_page:
                             <1>       ; 22/09/2015
                             <1>       ; 21/09/2015
                             <1>       ; 19/09/2015
                             <1>       ; 07/09/2015
                             <1>       ; 31/08/2015
                             <1>       ; 20/07/2015
                             <1>       ; 05/05/2015
                             <1>       ; 03/05/2015
                             <1>       ; 18/04/2015
                             <1>       ; 12/04/2015
                             <1>       ; 30/10/2014
                             <1>       ; 18/10/2014 (Retro UNIX 386 v1 - beginning)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      EBX = Virtual (linear) address of source page
                             <1>       ;           (Page fault address)
                             <1>       ; OUTPUT ->
                             <1>       ;      EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
                             <1>       ;      (corresponding PAGE TABLE ENTRY is mapped/set)
                             <1>       ;      EAX = 0 (CF = 1)
                             <1>       ;             if there is not a free page to be allocated
                             <1>       ;      (page content of the source page will be copied
                             <1>       ;      onto the target/new page)
                             <1>       ;
                             <1>       ; Modified Registers -> ecx, ebx (except EAX)
                             <1>       ;
00004F28 56                  <1>       push   esi
00004F29 57                  <1>       push   edi
                             <1>       ;push  ebx
                             <1>       ;push  ecx
00004F2A 31F6                <1>       xor    esi, esi
00004F2C C1EB0C              <1>       shr    ebx, 12 ; shift 12 bits right to get PDE & PTE numbers
00004F2F 89D9                <1>       mov    ecx, ebx ; save page fault address (as 12 bit shifted)
00004F31 C1EB08              <1>       shr    ebx, 8  ; shift 8 bits right and then
00004F34 80E3FC              <1>       and    bl, 0FCh ; mask lower 2 bits to get PDE offset
00004F37 89DF                <1>       mov    edi, ebx ; save it for the parent of current process
00004F39 031D[B8030300]      <1>       add    ebx, [u.pgdir] ; EBX points to the relevant page dir entry
00004F3F 8B03                <1>       mov    eax, [ebx] ; physical (base) address of the page table
00004F41 662500F0            <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
00004F45 89CB                <1>       mov    ebx, ecx   ; (restore higher 20 bits of page fault address)
00004F47 81E3FF030000        <1>       and    ebx, 3FFh  ; mask PDE# bits, the result is PTE# (0 to 1023)
00004F4D 66C1E302            <1>       shl    bx, 2      ; shift 2 bits left to get PTE offset
00004F51 01C3                <1>       add    ebx, eax   ; EBX points to the relevant page table entry
                             <1>       ; 07/09/2015
00004F53 66F7030002          <1>         test    word [ebx], PTE_DUPLICATED ; (Does current process share this
                             <1>                                ; read only page as a child process?)
00004F58 7509                <1>       jnz    short cpp_0 ; yes
00004F5A 8B0B                <1>       mov    ecx, [ebx] ; PTE value
00004F5C 6681E100F0          <1>       and    cx, PTE_A_CLEAR ; 0F000h  ; clear page attributes
00004F61 EB32                <1>       jmp    short cpp_1
                             <1> cpp_0:
00004F63 89FE                <1>       mov    esi, edi
00004F65 0335[BC030300]      <1>       add    esi, [u.ppgdir] ; the parent's page directory entry
00004F6B 8B06                <1>       mov    eax, [esi] ; physical (base) address of the page table
00004F6D 662500F0            <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
00004F71 89CE                <1>       mov    esi, ecx   ; (restore higher 20 bits of page fault address)
00004F73 81E6FF030000        <1>       and    esi, 3FFh  ; mask PDE# bits, the result is PTE# (0 to 1023)
00004F79 66C1E602            <1>       shl    si, 2      ; shift 2 bits left to get PTE offset
00004F7D 01C6                <1>       add    esi, eax   ; EDX points to the relevant page table entry
00004F7F 8B0E                <1>       mov    ecx, [esi] ; PTE value of the parent process
                             <1>       ; 21/09/2015
00004F81 8B03                <1>       mov    eax, [ebx] ; PTE value of the child process
00004F83 662500F0            <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear page attributes
```

```
                             <1>        ;
00004F87 F6C101              <1>        test   cl, PTE_A_PRESENT ; is it a present/valid page ?
00004F8A 7424                <1>        jz     short cpp_3 ; the parent's page is not same page
                             <1>        ;
00004F8C 6681E100F0          <1>        and    cx, PTE_A_CLEAR ; 0F000h ; clear page attributes
00004F91 39C8                <1>        cmp    eax, ecx   ; Same page?
00004F93 751B                <1>        jne    short cpp_3 ; Parent page and child page are not same
                             <1>                          ; Convert child's page to writable page
                             <1> cpp_1:
00004F95 E8E0FBFFFF          <1>        call   allocate_page
00004F9A 721A                <1>        jc     short cpp_4 ; 'insufficient memory' error
00004F9C 21F6                <1>        and    esi, esi    ; check ESI is valid or not
00004F9E 7405                <1>        jz     short cpp_2
                             <1>               ; Convert read only page to writable page
                             <1>               ;(for the parent of the current process)
                             <1>        ;and   word [esi], PTE_A_CLEAR ; 0F000h
                             <1>        ; 22/09/2015
00004FA0 890E                <1>        mov    [esi], ecx
00004FA2 800E07              <1>        or     byte [esi], PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER
                             <1>                              ; 1+2+4 = 7
                             <1> cpp_2:
00004FA5 89C7                <1>        mov    edi, eax ; new page address of the child process
                             <1>        ; 07/09/2015
00004FA7 89CE                <1>        mov    esi, ecx ; the page address of the parent process
00004FA9 B900040000          <1>        mov    ecx, PAGE_SIZE / 4
00004FAE F3A5                <1>        rep    movsd ; 31/08/2015
                             <1> cpp_3:
00004FB0 0C07                <1>        or     al, PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER ; 1+2+4 = 7
00004FB2 8903                <1>        mov    [ebx], eax ; Update PTE
00004FB4 28C0                <1>        sub    al, al ; clear attributes
                             <1> cpp_4:
                             <1>        ;pop   ecx
                             <1>        ;pop   ebx
00004FB6 5F                  <1>        pop    edi
00004FB7 5E                  <1>        pop    esi
00004FB8 C3                  <1>        retn
                             <1>
                             <1> ;; 28/04/2015
                             <1> ;; 24/10/2014
                             <1> ;; 21/10/2014 (Retro UNIX 386 v1 - beginning)
                             <1> ;; SWAP_PAGE_QUEUE (4096 bytes)
                             <1> ;;
                             <1> ;;   0000   0001   0002   0003   ....   1020   1021   1022   1023
                             <1> ;; +------+------+------+------+-     -+------+------+------+------+
                             <1> ;; |  pg1 |  pg2 |  pg3 |  pg4 | .... |pg1021|pg1022|pg1023|pg1024|
                             <1> ;; +------+------+------+------+-     -+------+------+------+------+
                             <1> ;;
                             <1> ;; [swpq_last] = 0 to 4096 (step 4) -> the last position on the queue
                             <1> ;;
                             <1> ;; Method:
                             <1> ;;    Swap page queue is a list of allocated pages with physical
                             <1> ;;    addresses (system mode virtual adresses = physical addresses).
                             <1> ;;    It is used for 'swap_in' and 'swap_out' procedures.
                             <1> ;;    When a new page is being allocated, swap queue is updated
                             <1> ;;    by 'swap_queue_shift' procedure, header of the queue (offset 0)
                             <1> ;;    is checked for 'accessed' flag. If the 1st page on the queue
                             <1> ;;    is 'accessed' or 'read only', it is dropped from the list;
                             <1> ;;    other pages from the 2nd to the last (in [swpq_last]) shifted
                             <1> ;;    to head then the 2nd page becomes the 1st and '[swpq_last]'
                             <1> ;;    offset value becomes it's previous offset value - 4.
                             <1> ;;    If the 1st page of the swap page queue is not 'accessed'
                             <1> ;;    the queue/list is not shifted.
                             <1> ;;    After the queue/list shift, newly allocated page is added
                             <1> ;;    to the tail of the queue at the [swpq_count*4] position.
                             <1> ;;    But, if [swpq_count] > 1023, the newly allocated page
                             <1> ;;    will not be added to the tail of swap page queue.
                             <1> ;;
                             <1> ;;    During 'swap_out' procedure, swap page queue is checked for
                             <1> ;;    the first non-accessed, writable page in the list,
                             <1> ;;    from the head to the tail. The list is shifted to left
                             <1> ;;    (to the head) till a non-accessed page will be found in the list.
                             <1> ;;    Then, this page     is swapped out (to disk) and then it is dropped
                             <1> ;;    from the list by a final swap queue shift. [swpq_count] value
                             <1> ;;    is changed. If all pages on the queue' are 'accessed',
                             <1> ;;    'insufficient memory' error will be returned ('swap_out'
                             <1> ;;    procedure will be failed)...
                             <1> ;;
                             <1> ;;    Note: If the 1st page of the queue is an 'accessed' page,
                             <1> ;;    'accessed' flag of the page will be reset (0) and that page
                             <1> ;;    (PTE) will be added to the tail of the queue after
                             <1> ;;    the check, if [swpq_count] < 1023. If [swpq_count] = 1024
                             <1> ;;    the queue will be rotated and the PTE in the head will be
                             <1> ;;    added to the tail after resetting 'accessed' bit.
                             <1> ;;
                             <1> ;;
                             <1> ;;
                             <1> ;; SWAP DISK/FILE (with 4096 bytes swapped page blocks)
                             <1> ;;
                             <1> ;;  00000000  00000004  00000008  0000000C   ...   size-8    size-4
                             <1> ;; +---------+---------+---------+---------+-- --+---------+---------+
                             <1> ;; |descriptr| page(1) | page(2) | page(3) | ... |page(n-1)| page(n) |
                             <1> ;; +---------+---------+---------+---------+-- --+---------+---------+
                             <1> ;;
                             <1> ;; [swpd_next] = the first free block address in swapped page records
                             <1> ;;                  for next free block search by 'swap_out' procedure.
                             <1> ;; [swpd_size] = swap disk/file size in sectors (512 bytes)
                             <1> ;;           NOTE: max. possible swap disk size is 1024 GB
                             <1> ;;           (entire swap space must be accessed by using
                             <1> ;;           31 bit offset address)
                             <1> ;; [swpd_free] = free block (4096 bytes) count in swap disk/file space
                             <1> ;; [swpd_start] = absolute/start address of the swap disk/file
                             <1> ;;            0 for file, or beginning sector of the swap partition
                             <1> ;; [swp_drv] = logical drive description table addr. of swap disk/file
```

```
<1> ;;
<1> ;;
<1> ;; Method:
<1> ;;    When the memory (ram) becomes insufficient, page allocation
<1> ;;    procedure swaps out a page from memory to the swap disk
<1> ;;    (partition) or swap file to get a new free page at the memory.
<1> ;;    Swapping out is performed by using swap page queue.
<1> ;;
<1> ;;    Allocation block size of swap disk/file is equal to page size
<1> ;;    (4096 bytes). Swapping address (in sectors) is recorded
<1> ;;    into relevant page file entry as 31 bit physical (logical)
<1> ;;    offset address as 1 bit shifted to left for present flag (0).
<1> ;;    Swapped page address is between 1 and swap disk/file size - 4.
<1> ;;    Absolute physical (logical) address of the swapped page is
<1> ;;    calculated by adding offset value to the swap partition's
<1> ;;    start address. If the swap device (disk) is a virtual disk
<1> ;;    or it is a file, start address of the swap disk/volume is 0,
<1> ;;    and offset value is equal to absolute (physical or logical)
<1> ;;    address/position. (It has not to be ZERO if the swap partition
<1> ;;    is in a partitioned virtual hard disk.)
<1> ;;
<1> ;;    Note: Swap addresses are always specified/declared in sectors,
<1> ;;    not in bytes or     in blocks/zones/clusters (4096 bytes) as unit.
<1> ;;
<1> ;;    Swap disk/file allocation is mapped via 'Swap Allocation Table'
<1> ;;    at memory as similar to 'Memory Allocation Table'.
<1> ;;
<1> ;;    Every bit of Swap Allocation Table repsesents one swap block
<1> ;;    (equal to page size) respectively. Bit 0 of the S.A.T. byte 0
<1> ;;    is reserved for swap disk/file block 0 as descriptor block
<1> ;;    (also for compatibility with PTE). If bit value is ZERO,
<1> ;;    it means relevant (respective) block is in use, and,
<1> ;;    of course, if bit value is 1, it means relevant (respective)
<1> ;;      swap disk/file block is free.
<1> ;;    For example: bit 1 of the byte 128 repsesents block 1025
<1> ;;    (128*8+1) or sector (offset) 8200 on the swap disk or
<1> ;;    byte (offset/position) 4198400 in the swap file.
<1> ;;    4GB swap space is represented via 128KB Swap Allocation Table.
<1> ;;    Initial layout of Swap Allocation Table is as follows:
<1> ;;    ------------------------------------------------------------
<1> ;;    0111111111111111111111111 .... 11111111111111111111111111111
<1> ;;    ------------------------------------------------------------
<1> ;;    (0 is reserved block, 1s represent free blocks respectively.)
<1> ;;    (Note: Allocation cell/unit of the table is bit, not byte)
<1> ;;
<1> ;;    ..............................................................
<1> ;;
<1> ;;    'swap_out' procedure checks 'free_swap_blocks' count at first,
<1> ;;    then it searches Swap Allocation Table if free count is not
<1> ;;    zero. From begining the [swpd_next] dword value, the first bit
<1> ;;    position with value of 1 on the table is converted to swap
<1> ;;    disk/file offset address, in sectors (not 4096 bytes block).
<1> ;;    'ldrv_write' procedure is called with ldrv (logical drive
<1> ;;    number of physical swap disk or virtual swap disk)
<1> ;;    number, sector offset (not absolute sector -LBA- number),
<1> ;;    and sector count (8, 512*8 = 4096) and buffer adress
<1> ;;    (memory page). That will be a direct disk write procedure.
<1> ;;    (for preventing late memory allocation, significant waiting).
<1> ;;    If disk write procedure returns with error or free count of
<1> ;;    swap blocks is ZERO, 'swap_out' procedure will return with
<1> ;;    'insufficient memory error' (cf=1).
<1> ;;
<1> ;;    (Note: Even if free swap disk/file blocks was not zero,
<1> ;;    any disk write error will not be fixed by 'swap_out' procedure,
<1> ;;    in other words, 'swap_out' will not check the table for other
<1> ;;    free blocks after a disk write error. It will return to
<1> ;;    the caller with error (CF=1) which means swapping is failed.
<1> ;;
<1> ;;    After writing the page on to swap disk/file address/sector,
<1> ;;    'swap_out' procesure returns with that swap (offset) sector
<1> ;;    address (cf=0).
<1> ;;
<1> ;;    ..............................................................
<1> ;;
<1> ;;    'swap_in' procedure loads addressed (relevant) swap disk or
<1> ;;    file sectors at specified memory page. Then page allocation
<1> ;;    procedure updates relevant page table entry with 'present'
<1> ;;    attribute. If swap disk or file reading fails there is nothing
<1> ;;    to do, except to terminate the process which is the owner of
<1> ;;    the swapped page.
<1> ;;
<1> ;;    'swap_in' procedure sets the relevant/respective bit value
<1> ;;    in the Swap Allocation Table (as free block). 'swap_in' also
<1> ;;    updates [swpd_first] pointer if it is required.
<1> ;;
<1> ;;    ..............................................................
<1> ;;
<1> ;;    Note: If [swap_enabled] value is ZERO, that means there is not
<1> ;;    a swap disk or swap file in use... 'swap_in' and 'swap_out'
<1> ;;    procedures ans 'swap page que' procedures will not be active...
<1> ;;    'Insufficient memory' error will be returned by 'swap_out'
<1> ;;    and 'general protection fault' will be returned by 'swap_in'
<1> ;;    procedure, if it is called mistakenly (a wrong value in a PTE).
<1> ;;
<1>
<1> swap_in:
<1>       ; 31/08/2015
<1>       ; 20/07/2015
<1>       ; 28/04/2015
<1>       ; 18/04/2015
<1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
<1>       ;
<1>       ; INPUT ->
```

```
                             <1>       ;      EBX = PHYSICAL (real/flat) ADDRESS OF THE MEMORY PAGE
                             <1>       ;      EBP = VIRTUAL (LINEAR) ADDRESS (page fault address)
                             <1>       ;      EAX = Offset Address for the swapped page on the
                             <1>       ;            swap disk or in the swap file.
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      EAX = 0 if loading at memory has been successful
                             <1>       ;
                             <1>       ;      CF = 1 -> swap disk reading error (disk/file not present
                             <1>       ;              or sector not present or drive not ready
                             <1>       ;           EAX = Error code
                             <1>       ;           [u.error] = EAX
                             <1>       ;                   = The last error code for the process
                             <1>       ;                     (will be reset after returning to user)
                             <1>       ;
                             <1>       ; Modified Registers -> EAX
                             <1>       ;
                             <1>
00004FB9 833D[62050300]00    <1>         cmp     dword [swp_drv], 0
00004FC0 7646                <1>       jna    short swpin_dnp_err
                             <1>
00004FC2 3B05[66050300]      <1>       cmp    eax, [swpd_size]
00004FC8 734A                <1>       jnb    short swpin_snp_err
                             <1>
00004FCA 56                  <1>       push   esi
00004FCB 53                  <1>       push   ebx
00004FCC 51                  <1>       push   ecx
00004FCD 8B35[62050300]      <1>       mov    esi, [swp_drv]
00004FD3 B908000000          <1>       mov    ecx, PAGE_SIZE / LOGIC_SECT_SIZE  ; 8 !
                             <1>              ; Note: Even if corresponding physical disk's sector
                             <1>              ; size different than 512 bytes, logical disk sector
                             <1>              ; size is 512 bytes and disk reading procedure
                             <1>              ; will be performed for reading 4096 bytes
                             <1>              ; (2*2048, 8*512).
                             <1>       ; ESI = Logical disk description table address
                             <1>       ; EBX = Memory page (buffer) address (physical!)
                             <1>       ; EAX = Sector adress (offset address, logical sector number)
                             <1>       ; ECX = Sector count ; 8 sectors
00004FD8 50                  <1>       push   eax
00004FD9 E8AF020000          <1>       call   logical_disk_read
00004FDE 58                  <1>       pop    eax
00004FDF 730C                <1>       jnc    short swpin_read_ok
                             <1>       ;
00004FE1 B828000000          <1>       mov    eax, SWP_DISK_READ_ERR ; drive not ready or read error
00004FE6 A3[C8030300]        <1>       mov    [u.error], eax
00004FEB EB17                <1>       jmp    short swpin_retn
                             <1>       ;
                             <1> swpin_read_ok:
                             <1>       ; EAX = Offset address (logical sector number)
00004FED E80D020000          <1>       call   unlink_swap_block  ; Deallocate swap block
                             <1>       ;
                             <1>       ; EBX = Memory page (buffer) address (physical!)
                             <1>       ; 20/07/2015
00004FF2 89EB                <1>       mov    ebx, ebp ; virtual address (page fault address)
00004FF4 6681E300F0          <1>         and     bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
00004FF9 8A1D[B3030300]      <1>       mov    bl, [u.uno] ; current process number
                             <1>       ; EBX = Virtual (Linear) address & process number combination
00004FFF E8DB000000          <1>       call   swap_queue_shift
                             <1>       ; eax = 0 ; 10/06/2016 (if ebx input > 0, eax output = 0)
                             <1>       ;sub   eax, eax  ; 0 ; Error Code = 0  (no error)
                             <1>       ; zf = 1
                             <1> swpin_retn:
00005004 59                  <1>       pop    ecx
00005005 5B                  <1>       pop    ebx
00005006 5E                  <1>       pop    esi
00005007 C3                  <1>       retn
                             <1>
                             <1> swpin_dnp_err:
00005008 B829000000          <1>       mov    eax, SWP_DISK_NOT_PRESENT_ERR
                             <1> swpin_err_retn:
0000500D A3[C8030300]        <1>       mov    [u.error], eax
00005012 F9                  <1>       stc
00005013 C3                  <1>       retn
                             <1>
                             <1> swpin_snp_err:
00005014 B82A000000          <1>       mov    eax, SWP_SECTOR_NOT_PRESENT_ERR
00005019 EBF2                <1>       jmp    short swpin_err_retn
                             <1>
                             <1> swap_out:
                             <1>       ; 10/06/2016
                             <1>       ; 07/06/2016
                             <1>         ; 23/05/2016
                             <1>       ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 24/10/2014 - 31/08/2015 (Retro UNIX 386 v1)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      none
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      EAX = Physical page address (which is swapped out
                             <1>       ;            for allocating a new page)
                             <1>       ;      CF = 1 -> swap disk writing error (disk/file not present
                             <1>       ;              or sector not present or drive not ready
                             <1>       ;           EAX = Error code
                             <1>       ;           [u.error] = EAX
                             <1>       ;                   = The last error code for the process
                             <1>       ;                     (will be reset after returning to user)
                             <1>       ;
                             <1>       ; Modified Registers -> none (except EAX)
                             <1>       ;
0000501B 66833D[60050300]01  <1>       cmp    word [swpq_count], 1
00005023 0F82AF000000        <1>         jc      swpout_im_err ; 'insufficient memory'
                             <1>
                             <1>         ;cmp    dword [swp_drv], 1
                             <1>       ;jc    short swpout_dnp_err ; 'swap disk/file not present'
                             <1> 
00005029 833D[6A050300]01    <1>         cmp     dword [swpd_free], 1
00005030 0F828F000000        <1>         jc      swpout_nfspc_err ; 'no free space on swap disk'
                             <1> 
00005036 53                  <1>       push   ebx ; *
                             <1> swpout_1:
                             <1>       ; 10/06/2016
00005037 31DB                <1>       xor    ebx, ebx ; shift the queue and return a PTE value
00005039 E8A1000000          <1>       call   swap_queue_shift
0000503E 21C0                <1>       and    eax, eax    ; 0 = empty queue (improper entries)
00005040 0F848A000000        <1>         jz      swpout_npts_err        ; There is not any proper PTE
                             <1>                                 ; pointer in the swap queue
                             <1>       ; EAX = PTE value of the page
                             <1>       ; EBX = PTE address of the page
00005046 662500F0            <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
                             <1>       ;
                             <1>       ; 07/06/2016
                             <1>       ; 19/05/2016
                             <1>       ; check this page is in timer events or not
                             <1> 
                             <1> swpout_timer_page_0:
0000504A 52                  <1>       push   edx ; **
                             <1> 
                             <1>       ; 07/06/2016
0000504B 803D[CF650100]00    <1>       cmp    byte [timer_events], 0
00005052 762F                <1>       jna    short swpout_2
                             <1>       ;
00005054 8A15[CF650100]      <1>       mov    dl, [timer_events]
                             <1> 
0000505A 51                  <1>       push   ecx ; ***
0000505B 53                  <1>       push   ebx ; ****
0000505C BB[60040300]        <1>       mov    ebx, timer_set ; beginning address of timer event
                             <1>                           ; structures
                             <1> swpout_timer_page_1:
00005061 8A0B                <1>       mov    cl, [ebx]
00005063 08C9                <1>       or     cl, cl ; 0 = free, >0 = process number
00005065 7415                <1>       jz     short swpout_timer_page_3
00005067 8B4B0C              <1>       mov    ecx, [ebx+12] ; response (signal return) address
0000506A 6681E100F0          <1>       and    cx, PTE_A_CLEAR ; clear offset part (right 12 bits)
                             <1>                          ; of the response byte address, to
                             <1>                          ; get beginning of the page address)
0000506F 39C8                <1>       cmp    eax, ecx
00005071 7505                <1>       jne    short swpout_timer_page_2 ; not same page
                             <1> 
                             <1>       ; !same page!
                             <1>       ;
                             <1>       ; NOTE: // 19/05/2016 // - TRDOS 386 feature only ! -
                             <1>       ; This page will be used by the kernel to put timer event
                             <1>       ; response (signal return) byte at the requested address;
                             <1>       ; in order to prevent a possible wrong write (while
                             <1>       ; this page is swapped out) on physical memory,
                             <1>       ; we must protect this page against to be swapped out!
                             <1>       ;
00005073 5B                  <1>       pop    ebx ; ****
00005074 59                  <1>       pop    ecx ; ***
00005075 5A                  <1>       pop    edx ; **
00005076 EBBF                <1>       jmp    short swpout_1     ; do not swap out this page !
                             <1> 
                             <1> swpout_timer_page_2:
                             <1>       ; 07/06/2016
00005078 FECA                <1>       dec    dl
0000507A 7405                <1>       jz     short swpout_timer_page_4
                             <1> swpout_timer_page_3:
                             <1>       ;cmp   ebx, timer_set + 240 ; last timer event (15*16)
                             <1>       ;jnb   short swpout_timer_page_4
0000507C 83C310              <1>       add    ebx, 16
0000507F EBE0                <1>       jmp    short swpout_timer_page_1
                             <1> 
                             <1> swpout_timer_page_4:
00005081 5B                  <1>       pop    ebx ; ****
00005082 59                  <1>       pop    ecx ; ***
                             <1> swpout_2:
00005083 89DA                <1>       mov    edx, ebx           ; Page table entry address
00005085 89C3                <1>       mov    ebx, eax           ; Buffer (Page) Address
                             <1>       ;
00005087 E8A6010000          <1>       call   link_swap_block
0000508C 7304                <1>       jnc    short swpout_3            ; It may not be needed here
                             <1>                                 ; because [swpd_free] value
                             <1>                                 ; was checked at the beginging.
0000508E 5A                  <1>       pop    edx ; **
0000508F 5B                  <1>       pop    ebx ; *
00005090 EB33                <1>       jmp    short swpout_nfspc_err
                             <1> swpout_3:
00005092 A900000080          <1>       test   eax, 80000000h ; test bit 31 (this may not be needed!)
00005097 752C                <1>       jnz    short swpout_nfspc_err  ; 10/06/2016 (bit 31 = 1 !)
                             <1>       ;
00005099 56                  <1>       push   esi ; **
0000509A 51                  <1>       push   ecx ; ***
0000509B 50                  <1>       push   eax ; sector address ; (31 bit !, bit 31 = 0)
0000509C 8B35[62050300]      <1>       mov    esi, [swp_drv]
000050A2 B908000000          <1>       mov    ecx, PAGE_SIZE / LOGIC_SECT_SIZE  ; 8 !
                             <1>             ; Note: Even if corresponding physical disk's sector
                             <1>             ; size different than 512 bytes, logical disk sector
                             <1>             ; size is 512 bytes and disk writing procedure
                             <1>             ; will be performed for writing 4096 bytes
                             <1>             ; (2*2048, 8*512).
                             <1>       ; ESI = Logical disk description table address
                             <1>       ; EBX = Buffer (Page) address
                             <1>       ; EAX = Sector adress (offset address, logical sector number)
                             <1>       ; ECX = Sector count ; 8 sectors
                             <1>       ; edx = PTE address
```

```
000050A7 E8E2010000          <1>       call   logical_disk_write
                             <1>       ; edx = PTE address
000050AC 59                  <1>       pop    ecx ; sector address
000050AD 730C                <1>       jnc    short swpout_write_ok
                             <1>       ;
                             <1>       ;; call      unlink_swap_block ; this block must be left as 'in use'
                             <1> swpout_dw_err:
000050AF B82C000000          <1>       mov    eax, SWP_DISK_WRITE_ERR ; drive not ready or write error
000050B4 A3[C8030300]        <1>       mov    [u.error], eax
000050B9 EB06                <1>       jmp    short swpout_retn
                             <1>       ;
                             <1> swpout_write_ok:
                             <1>       ; EBX = Buffer (page) address
                             <1>       ; EDX = Page Table Entry address
                             <1>       ; ECX = Swap disk sector (file block) address (31 bit)
000050BB D1E1                <1>       shl    ecx, 1  ; 31 bit sector address from bit 1 to bit 31
000050BD 890A                <1>       mov    [edx], ecx
                             <1>              ; bit 0 = 0 (swapped page)
000050BF 89D8                <1>       mov    eax, ebx
                             <1> swpout_retn:
000050C1 59                  <1>       pop    ecx ; ***
000050C2 5E                  <1>       pop    esi ; **
000050C3 5B                  <1>       pop    ebx ; *
000050C4 C3                  <1>       retn
                             <1>
                             <1> ;swpout_dnp_err:
                             <1> ;     mov    eax, SWP_DISK_NOT_PRESENT_ERR ; disk not present
                             <1> ;     jmp    short swpout_err_retn
                             <1> swpout_nfspc_err:
000050C5 B82B000000          <1>       mov    eax, SWP_NO_FREE_SPACE_ERR ; no free space
                             <1> swpout_err_retn:
000050CA A3[C8030300]        <1>       mov    [u.error], eax
                             <1>       ;stc
000050CF C3                  <1>       retn
                             <1> swpout_npts_err:
000050D0 B82D000000          <1>       mov    eax, SWP_NO_PAGE_TO_SWAP_ERR
000050D5 5B                  <1>       pop    ebx
000050D6 EBF2                <1>       jmp    short swpout_err_retn
                             <1> swpout_im_err:
000050D8 B804000000          <1>       mov    eax, ERR_MINOR_IM ; insufficient (out of) memory
000050DD EBEB                <1>       jmp    short swpout_err_retn
                             <1>
                             <1> swap_queue_shift:
                             <1>       ; 26/03/2017
                             <1>       ; 10/06/2016
                             <1>       ; 09/06/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 23/10/2014 - 20/07/2015 (Retro UNIX 386 v1)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      EBX = Virtual (linear) address (bit 12 to 31)
                             <1>       ;            and process number combination (bit 0 to 11)
                             <1>       ;      EBX = 0 -> shift/drop from the head (offset 0)
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      If EBX input > 0
                             <1>       ;         the queue will be shifted 4 bytes (dword),
                             <1>       ;         from the tail to the head, up to entry offset
                             <1>       ;         which points to EBX input value or nothing
                             <1>       ;         to do if EBX value is not found on the queue.
                             <1>       ;         (The entry -with EBX value- will be removed
                             <1>       ;         from the queue if it is found.)
                             <1>       ;
                             <1>       ;         EAX = 0
                             <1>       ;
                             <1>       ;      If EBX input = 0
                             <1>       ;         the queue will be shifted 4 bytes (dword),
                             <1>       ;         from the tail to the head, if the PTE address
                             <1>       ;         which is pointed in head of the queue is marked
                             <1>       ;         as "accessed" or it is marked as "non present".
                             <1>       ;         (If "accessed" flag of the PTE -which is pointed
                             <1>       ;         in the head- is set -to 1-, it will be reset
                             <1>       ;         -to 0- and then, the queue will be rotated
                             <1>       ;         -without dropping pointer of the PTE from
                             <1>       ;         the queue- for 4 bytes on head to tail direction.
                             <1>       ;         Pointer in the head will be moved into the tail,
                             <1>       ;         other PTEs will be shifted on head direction.)
                             <1>       ;
                             <1>       ;         Swap queue will be shifted up to the first
                             <1>       ;         'present' or 'non accessed' page will be found
                             <1>       ;         (as pointed) on the queue head (then it will be
                             <1>         ;          removed/dropped from the queue).
                             <1>       ;
                             <1>       ;         EAX (> 0) = PTE value of the page which is
                             <1>       ;              (it's pointer -virtual address-) dropped
                             <1>       ;              (removed) from swap queue.
                             <1>       ;         EBX = PTE address of the page (if EAX > 0)
                             <1>       ;               which is (it's pointer -virtual address-)
                             <1>       ;              dropped (removed) from swap queue.
                             <1>       ;
                             <1>       ;         EAX = 0 -> empty swap queue !
                             <1>       ;
                             <1>       ; Modified Registers -> EAX, EBX
                             <1>       ;
000050DF 0FB705[60050300]    <1>       movzx   eax, word [swpq_count]  ; Max. 1024
000050E6 6621C0              <1>       and    ax, ax
000050E9 7431                <1>       jz     short swpqs_retn
000050EB 57                  <1>       push   edi
000050EC 56                  <1>       push   esi
000050ED 51                  <1>       push   ecx
000050EE BE00E00800          <1>       mov    esi, swap_queue
000050F3 89C1                <1>       mov    ecx, eax
000050F5 09DB                <1>       or     ebx, ebx
000050F7 7424                <1>       jz     short swpqs_7
```

```
                             <1> swpqs_1:
000050F9 AD                  <1>       lodsd
000050FA 39D8                <1>       cmp    eax, ebx
000050FC 7406                <1>       je     short swpqs_2
000050FE E2F9                <1>       loop   swpqs_1
                             <1>       ; 10/06/2016
00005100 29C0                <1>       sub    eax, eax
00005102 EB15                <1>       jmp    short swpqs_6
                             <1> swpqs_2:
00005104 89F7                <1>       mov    edi, esi
00005106 83EF04              <1>       sub    edi, 4
                             <1> swpqs_3:
00005109 66FF0D[60050300]    <1>       dec    word [swpq_count]
00005110 7403                <1>       jz     short swpqs_5
                             <1> swpqs_4:
00005112 49                  <1>       dec    ecx
00005113 F3A5                <1>       rep    movsd  ; shift up (to the head)
                             <1> swpqs_5:
00005115 31C0                <1>       xor    eax, eax
00005117 8907                <1>       mov    [edi], eax
                             <1> swpqs_6:
00005119 59                  <1>       pop    ecx
0000511A 5E                  <1>       pop    esi
0000511B 5F                  <1>       pop    edi
                             <1> swpqs_retn:
0000511C C3                  <1>       retn
                             <1> swpqs_7:
0000511D 89F7                <1>       mov    edi, esi ; head
0000511F AD                  <1>       lodsd
                             <1>       ; 20/07/2015
00005120 89C3                <1>       mov    ebx, eax
00005122 81E300F0FFFF        <1>       and    ebx, ~PAGE_OFF ; ~0FFFh
                             <1>                    ; ebx = virtual address (at page boundary)
00005128 25FF0F0000          <1>       and    eax, PAGE_OFF ; 0FFFh
                             <1>                    ; ax = process number (1 to 4095)
0000512D 3A05[B3030300]      <1>       cmp    al, [u.uno]
                             <1>              ; Max. 16 (nproc) processes for Retro UNIX 386 v1
00005133 7507                <1>       jne    short swpqs_8
00005135 A1[B8030300]        <1>       mov    eax, [u.pgdir]
0000513A EB28                <1>       jmp    short swpqs_9
                             <1> swpqs_8:
                             <1>       ; 09/06/2016
0000513C 80B8[AF000300]00    <1>       cmp    byte [eax+p.stat-1], 0
00005143 76C4                <1>       jna    short swpqs_3     ; free (or terminated) process
00005145 80B8[AF000300]02    <1>       cmp    byte [eax+p.stat-1], 2 ; waiting
0000514C 77BB                <1>       ja     short swpqs_3         ; zombie (3) or undefined ?
                             <1>
                             <1>       ;shl   ax, 2
0000514E C0E002              <1>       shl    al, 2
00005151 8B80[BC000300]      <1>       mov    eax, [eax+p.upage-4]
00005157 09C0                <1>       or     eax, eax
00005159 74AE                <1>       jz     short swpqs_3 ; invalid upage
0000515B 83C05C              <1>       add    eax, u.pgdir - user
                             <1>                     ; u.pgdir value for the process
                             <1>                     ; is in [eax]
0000515E 8B00                <1>       mov    eax, [eax]
00005160 21C0                <1>       and    eax, eax
00005162 74A5                <1>       jz     short swpqs_3 ; invalid page directory
                             <1> swpqs_9:
00005164 52                  <1>       push   edx
                             <1>       ; eax = page directory
                             <1>       ; ebx = virtual address
00005165 E82BFBFFFF          <1>       call   get_pte
0000516A 89D3                <1>       mov    ebx, edx     ; PTE address
0000516C 5A                  <1>       pop    edx
                             <1>       ; 10/06/2016
0000516D 723A                <1>       jc     short swpqs_13 ; empty PDE
                             <1>       ; EAX = PTE value
0000516F A801                <1>       test   al, PTE_A_PRESENT ; bit 0 = 1
00005171 7436                <1>       jz     short swpqs_13  ; Drop non-present page
                             <1>                              ; from the queue (head)
00005173 A802                <1>       test   al, PTE_A_WRITE     ; bit 1 = 0 (read only)
00005175 7432                <1>       jz     short swpqs_13  ; Drop read only page
                             <1>                              ; from the queue (head)
                             <1>       ;test  al, PTE_A_ACCESS ; bit 5 = 1 (Accessed)
                             <1>       ;jnz   short swpqs_11  ; present
                             <1>                              ; accessed page
00005177 0FBAF005            <1>         btr     eax, PTE_A_ACCESS_BIT ; reset 'accessed' bit
0000517B 7210                <1>       jc     short swpqs_11  ; accessed page
                             <1>
0000517D 49                  <1>       dec    ecx
0000517E 66890D[60050300]    <1>       mov    [swpq_count], cx
00005185 7402                <1>         jz      short swpqs_10
                             <1>              ; esi = head + 4
                             <1>              ; edi = head
00005187 F3A5                <1>       rep    movsd   ; n = 1 to k-1, [n - 1] = [n]
                             <1> swpqs_10:
00005189 890F                <1>       mov    [edi], ecx ; 0
0000518B EB8C                <1>       jmp    short swpqs_6 ; 26/03/2017
                             <1>
                             <1> swpqs_11:
0000518D 8903                <1>       mov    [ebx], eax     ; save changed attribute
                             <1>       ; Rotation (head -> tail)
0000518F 49                  <1>       dec    ecx     ; entry count -> last entry number
00005190 74F7                <1>       jz     short swpqs_10
                             <1>              ; esi = head + 4
                             <1>              ; edi = head
00005192 8B07                <1>       mov    eax, [edi] ; 20/07/2015
00005194 F3A5                <1>       rep    movsd   ; n = 1 to k-1, [n - 1] = [n]
00005196 8907                <1>       mov    [edi], eax ; head -> tail ; [k] = [1]
                             <1>
00005198 668B0D[60050300]    <1>       mov    cx, [swpq_count]
                             <1>
                             <1> swpqs_12:
0000519F BE00E00800          <1>       mov    esi, swap_queue ; head
000051A4 E974FFFFFF          <1>         jmp     swpqs_7
                             <1>
                             <1> swpqs_13:
000051A9 49                  <1>       dec    ecx
000051AA 66890D[60050300]    <1>       mov    [swpq_count], cx
000051B1 0F845EFFFFFF        <1>         jz      swpqs_5
000051B7 EBE6                <1>       jmp    short swpqs_12
                             <1>
                             <1> add_to_swap_queue:
                             <1> ; temporary - 16/09/2015
000051B9 C3                  <1> retn
                             <1>       ; 20/02/2017
                             <1>       ; 20/07/2015
                             <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
                             <1>       ;
                             <1>       ; Adds new page to swap queue
                             <1>       ; (page directories and page tables must not be added
                             <1>       ; to swap queue)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      EBX = Linear (Virtual) addr for current process
                             <1>       ;      [u.uno]
                             <1>       ;      20/02/2017
                             <1>       ;      (Linear address = CORE + user's virtual address)
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      EAX = [swpq_count]
                             <1>       ;            (after the PTE has been added)
                             <1>       ;      EAX = 0 -> Swap queue is full, (1024 entries)
                             <1>       ;            the PTE could not be added.
                             <1>       ;
                             <1>       ; Modified Registers -> EAX
                             <1>       ;
000051BA 53                  <1>       push   ebx
000051BB 6681E300F0          <1>         and     bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
000051C0 8A1D[B3030300]      <1>       mov    bl, [u.uno] ; current process number
000051C6 E814FFFFFF          <1>       call   swap_queue_shift ; drop from the queue if
                             <1>                          ; it is already on the queue
                             <1>             ; then add it to the tail of the queue
000051CB 0FB705[60050300]    <1>       movzx  eax, word [swpq_count]
000051D2 663D0004            <1>       cmp    ax, 1024
000051D6 7205                <1>       jb     short atsq_1
000051D8 6629C0              <1>       sub    ax, ax
000051DB 5B                  <1>       pop    ebx
000051DC C3                  <1>       retn
                             <1> atsq_1:
000051DD 56                  <1>       push   esi
000051DE BE00E00800          <1>       mov    esi, swap_queue
000051E3 6621C0              <1>       and    ax, ax
000051E6 740A                <1>       jz     short atsq_2
000051E8 66C1E002            <1>       shl    ax, 2  ; convert to offset
000051EC 01C6                <1>       add    esi, eax
000051EE 66C1E802            <1>       shr    ax, 2
                             <1> atsq_2:
000051F2 6640                <1>       inc    ax
000051F4 891E                <1>       mov    [esi], ebx ; Virtual address + [u.uno] combination
000051F6 66A3[60050300]      <1>       mov    [swpq_count], ax
000051FC 5E                  <1>       pop    esi
000051FD 5B                  <1>       pop    ebx
000051FE C3                  <1>       retn
                             <1>
                             <1> unlink_swap_block:
                             <1>       ; 15/09/2015
                             <1>       ; 30/04/2015
                             <1>       ; 18/04/2015
                             <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      EAX = swap disk/file offset address
                             <1>       ;            (bit 1 to bit 31)
                             <1>       ; OUTPUT ->
                             <1>       ;      [swpd_free] is increased
                             <1>       ;      (corresponding SWAP DISK ALLOC. TABLE bit is SET)
                             <1>       ;
                             <1>       ; Modified Registers -> EAX
                             <1>       ;
000051FF 53                  <1>       push   ebx
00005200 52                  <1>       push   edx
                             <1>       ;
00005201 C1E804              <1>       shr    eax, SECTOR_SHIFT+1  ;3+1 ; shift sector address to
                             <1>                               ; 3 bits right
                             <1>                               ; to get swap block/page number
00005204 89C2                <1>       mov    edx, eax
                             <1>       ; 15/09/2015
00005206 C1EA03              <1>       shr    edx, 3            ; to get offset to S.A.T.
                             <1>                               ; (1 allocation bit = 1 page)
                             <1>                               ; (1 allocation bytes = 8 pages)
00005209 80E2FC              <1>       and    dl, 0FCh          ; clear lower 2 bits
                             <1>                               ; (to get 32 bit position)
                             <1>       ;
0000520C BB00000D00          <1>       mov    ebx, swap_alloc_table ; Swap Allocation Table address
00005211 01D3                <1>       add    ebx, edx
00005213 83E01F              <1>       and    eax, 1Fh          ; lower 5 bits only
                             <1>                               ; (allocation bit position)
00005216 3B05[6E050300]      <1>       cmp    eax, [swpd_next]     ; is the new free block addr. lower
                             <1>                               ; than the address in 'swpd_next' ?
                             <1>                               ; (next/first free block value)
0000521C 7305                <1>       jnb    short uswpbl_1           ; no
0000521E A3[6E050300]        <1>       mov    [swpd_next], eax     ; yes
                             <1> uswpbl_1:
00005223 0FAB03              <1>       bts    [ebx], eax        ; unlink/release/deallocate block
```

```
                             <1>                               ; set relevant bit to 1.
                             <1>                               ; set CF to the previous bit value
00005226 F5                  <1>       cmc                     ; complement carry flag
00005227 7206                <1>       jc     short uswpbl_2          ; do not increase swfd_free count
                             <1>                               ; if the block is already deallocated
                             <1>                               ; before.
00005229 FF05[6A050300]      <1>         inc     dword [swpd_free]
                             <1> uswpbl_2:
0000522F 5A                  <1>       pop    edx
00005230 5B                  <1>       pop    ebx
00005231 C3                  <1>       retn
                             <1>
                             <1> link_swap_block:
                             <1>       ; 01/07/2015
                             <1>       ; 18/04/2015
                             <1>       ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
                             <1>       ;
                             <1>       ; INPUT -> none
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      EAX = OFFSET ADDRESS OF THE ALLOCATED BLOCK (4096 bytes)
                             <1>       ;            in sectors (corresponding
                             <1>       ;            SWAP DISK ALLOCATION TABLE bit is RESET)
                             <1>       ;
                             <1>       ;      CF = 1 and EAX = 0
                             <1>       ;               if there is not a free block to be allocated
                             <1>       ;
                             <1>       ; Modified Registers -> none (except EAX)
                             <1>       ;
                             <1>
                             <1>       ;mov   eax, [swpd_free]
                             <1>       ;and   eax, eax
                             <1>       ;jz    short out_of_swpspc
                             <1>       ;
00005232 53                  <1>       push   ebx
00005233 51                  <1>       push   ecx
                             <1>       ;
00005234 BB00000D00          <1>       mov    ebx, swap_alloc_table ; Swap Allocation Table offset
00005239 89D9                <1>       mov    ecx, ebx
0000523B 031D[6E050300]      <1>       add    ebx, [swpd_next] ; Free block searching starts from here
                             <1>                           ; next_free_swap_block >> 5
00005241 030D[72050300]      <1>       add    ecx, [swpd_last] ; Free block searching ends here
                             <1>                           ; (total_swap_blocks - 1) >> 5
                             <1> lswbl_scan:
00005247 39CB                <1>       cmp    ebx, ecx
00005249 770A                <1>       ja     short lswbl_notfound
                             <1>       ;
0000524B 0FBC03              <1>       bsf    eax, [ebx] ; Scans source operand for first bit set (1).
                             <1>                      ; Clears ZF if a bit is found set (1) and
                             <1>                      ; loads the destination with an index to
                             <1>                      ; first set bit. (0 -> 31)
                             <1>                      ; Sets ZF to 1 if no bits are found set.
                             <1>       ; 01/07/2015
0000524E 751C                <1>       jnz    short lswbl_found ; ZF = 0 -> a free block has been found
                             <1>                    ;
                             <1>                    ; NOTE:  a Swap Disk Allocation Table bit
                             <1>                    ;       with value of 1 means
                             <1>                    ;       the corresponding page is free
                             <1>                    ;       (Retro UNIX 386 v1 feaure only!)
00005250 83C304              <1>       add    ebx, 4
                             <1>                    ; We return back for searching next page block
                             <1>                    ; NOTE: [swpd_free] is not ZERO; so,
                             <1>                    ;      we always will find at least 1 free block here.
00005253 EBF2                <1>       jmp          short lswbl_scan
                             <1>       ;
                             <1> lswbl_notfound:
00005255 81E900000D00        <1>       sub    ecx, swap_alloc_table
0000525B 890D[6E050300]      <1>       mov    [swpd_next], ecx ; next/first free page = last page
                             <1>                           ; (unlink_swap_block procedure will change it)
00005261 31C0                <1>       xor    eax, eax
00005263 A3[6A050300]        <1>       mov    [swpd_free], eax
00005268 F9                  <1>       stc
                             <1> lswbl_ok:
00005269 59                  <1>       pop    ecx
0000526A 5B                  <1>       pop    ebx
0000526B C3                  <1>       retn
                             <1>       ;
                             <1> ;out_of_swpspc:
                             <1> ;     stc
                             <1> ;     retn
                             <1>
                             <1> lswbl_found:
0000526C 89D9                <1>       mov    ecx, ebx
0000526E 81E900000D00        <1>       sub    ecx, swap_alloc_table
00005274 890D[6E050300]      <1>       mov    [swpd_next], ecx ; Set first free block searching start
                             <1>                           ; address/offset (to the next)
0000527A FF0D[6A050300]      <1>         dec     dword [swpd_free] ; 1 block has been allocated (X = X-1)
                             <1>       ;
00005280 0FB303              <1>       btr    [ebx], eax    ; The destination bit indexed by the source value
                             <1>                           ; is copied into the Carry Flag and then cleared
                             <1>                           ; in the destination.
                             <1>                           ;
                             <1>                           ; Reset the bit which is corresponding to the
                             <1>                           ; (just) allocated block.
00005283 C1E105              <1>       shl    ecx, 5        ; (block offset * 32) + block index
00005286 01C8                <1>       add    eax, ecx      ; = block number
00005288 C1E003              <1>       shl    eax, SECTOR_SHIFT ; 3, sector (offset) address of the block
                             <1>                           ; 1 block =  8 sectors
                             <1>       ;
                             <1>       ; EAX = offset address of swap disk/file sector (beginning of the block)
                             <1>       ;
                             <1>       ; NOTE: The relevant page table entry will be updated
                             <1>       ;       according to this EAX value...
                             <1>        ;
0000528B EBDC                <1>        jmp    short lswbl_ok
                             <1>
                             <1> logical_disk_read:
                             <1>        ; 20/07/2015
                             <1>        ; 09/03/2015 (temporary code here)
                             <1>        ;
                             <1>        ; INPUT ->
                             <1>        ;      ESI = Logical disk description table address
                             <1>        ;      EBX = Memory page (buffer) address (physical!)
                             <1>        ;      EAX = Sector adress (offset address, logical sector number)
                             <1>        ;      ECX = Sector count
                             <1>        ;
                             <1>        ;
0000528D C3                  <1>        retn
                             <1>
                             <1> logical_disk_write:
                             <1>        ; 20/07/2015
                             <1>        ; 09/03/2015 (temporary code here)
                             <1>        ;
                             <1>        ; INPUT ->
                             <1>        ;      ESI = Logical disk description table address
                             <1>        ;      EBX = Memory page (buffer) address (physical!)
                             <1>        ;      EAX = Sector adress (offset address, logical sector number)
                             <1>        ;      ECX = Sector count
                             <1>        ;
0000528E C3                  <1>        retn
                             <1>
                             <1> get_physical_addr:
                             <1>        ; 26/03/2017
                             <1>        ; 20/02/2017
                             <1>        ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>        ; 18/10/2015
                             <1>        ; 29/07/2015
                             <1>        ; 20/07/2015
                             <1>        ; 04/06/2015
                             <1>        ; 20/05/2015
                             <1>        ; 28/04/2015
                             <1>        ; 18/04/2015
                             <1>        ; Get physical address
                             <1>        ;     (allocates a new page for user if it is not present)
                             <1>        ;
                             <1>        ; (This subroutine is needed for mapping user's virtual
                             <1>        ; (buffer) address to physical address (of the buffer).)
                             <1>        ; ('sys write', 'sys read' system calls...)
                             <1>        ;
                             <1>        ; INPUT ->
                             <1>        ;      EBX = virtual address
                             <1>        ;      u.pgdir = page directory (physical) address
                             <1>        ;
                             <1>        ; OUTPUT ->
                             <1>        ;      EAX = physical address
                             <1>        ;      EBX = linear address
                             <1>        ;      EDX = physical address of the page frame
                             <1>        ;            (with attribute bits)
                             <1>        ;      ECX = byte count within the page frame
                             <1>        ;
                             <1>        ; Modified Registers -> EAX, EBX, ECX, EDX
                             <1>        ;
0000528F 81C300004000        <1>        add    ebx, CORE ; 18/10/2015
                             <1> get_physical_addr_x: ; 27/05/2016
00005295 A1[B8030300]        <1>        mov    eax, [u.pgdir]
0000529A E8F6F9FFFF          <1>        call   get_pte
                             <1>               ; EDX = Page table entry address (if CF=0)
                             <1>                 ;       Page directory entry address (if CF=1)
                             <1>               ;       (Bit 0 value is 0 if PT is not present)
                             <1>               ; EAX = Page table entry value (page address)
                             <1>               ;      CF = 1 -> PDE not present or invalid ?
0000529F 731C                <1>        jnc    short gpa_1
                             <1>        ;
000052A1 E8D4F8FFFF          <1>        call   allocate_page
000052A6 7248                <1>        jc     short gpa_im_err  ; 'insufficient memory' error
                             <1> gpa_0:
000052A8 E847F9FFFF          <1>        call   clear_page
                             <1>        ; EAX = Physical (base) address of the allocated (new) page
000052AD 0C07                <1>        or     al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER ; 4+2+1 = 7
                             <1>                         ; lower 3 bits are used as U/S, R/W, P flags
                             <1>                         ; (user, writable, present page)
000052AF 8902                <1>        mov    [edx], eax ; Let's put the new page directory entry here !
000052B1 A1[B8030300]        <1>        mov    eax, [u.pgdir]
000052B6 E8DAF9FFFF          <1>        call   get_pte
000052BB 7233                <1>        jc     short gpa_im_err ; 'insufficient memory' error
                             <1> gpa_1:
                             <1>        ; EAX = PTE value, EDX = PTE address
000052BD A801                <1>        test   al, PTE_A_PRESENT
000052BF 751F                <1>        jnz    short gpa_3 ; 26/03/2017
000052C1 09C0                <1>        or     eax, eax
000052C3 7456                <1>        jz     short gpa_7  ; Allocate a new page
                             <1>        ; 20/07/2015
000052C5 55                  <1>        push   ebp
000052C6 89DD                <1>        mov    ebp, ebx ; virtual (linear) address
                             <1>        ; reload swapped page
000052C8 E878000000          <1>        call   reload_page ; 28/04/2015
000052CD 5D                  <1>        pop    ebp
000052CE 724A                <1>        jc     short gpa_retn
                             <1> gpa_2:
                             <1>        ; 26/03/2017
                             <1>        ; 20/02/2017
                             <1>        ; If a page will contain a Signal Response Byte
                             <1>        ; it must not be swapped out, because
                             <1>        ; timer service or irq callback service
                             <1>        ; will write a signal return/response byte
                             <1>        ; directly by using physical address of Signal
                             <1>       ; Response Byte.(Even if process is not running,
                             <1>       ; or it is running with swapped out pages.)
                             <1>       ;
                             <1>       ; 'no_page_swap' will be set by 'systimer' or
                             <1>       ; 'syscalbac' sistem functions/calls. (*)
                             <1>       ;
000052D0 803D[0E6B0100]00    <1>       cmp    byte [no_page_swap], 0
000052D7 761D                <1>       jna    short gpa_4 ; this page can be swapped out
                             <1>       ; this page must not be swapped out
                             <1>       ; but 'no_page_swap' must be reset here
                             <1>       ; imediately for other callers (*)
                             <1>       ; (otherwise, swap queue would not be long enough)
000052D9 E84B000000          <1>       call   gpa_8 ; 26/03/2017
000052DE EB1D                <1>       jmp    short gpa_5
                             <1> gpa_3:
                             <1>       ; 26/03/2017
000052E0 803D[0E6B0100]00    <1>       cmp    byte [no_page_swap], 0
000052E7 7618                <1>       jna    short gpa_6 ; this page can be swapped out
000052E9 E83B000000          <1>       call   gpa_8
000052EE EB11                <1>       jmp    short gpa_6
                             <1>
                             <1> gpa_im_err:
000052F0 B804000000          <1>       mov    eax, ERR_MINOR_IM ; Insufficient memory (minor) error!
                             <1>                             ; Major error = 0 (No protection fault)
000052F5 C3                  <1>       retn
                             <1> gpa_4:
                             <1>       ; 20/07/2015
                             <1>       ; 20/05/2015
                             <1>       ; add this page to swap queue
000052F6 50                  <1>       push   eax
                             <1>       ; EBX = Linear (CORE+virtual) address ; 20/02/2017
000052F7 E8BDFEFFFF          <1>       call   add_to_swap_queue
000052FC 58                  <1>       pop    eax
                             <1> gpa_5:
                             <1>              ; PTE address in EDX
                             <1>              ; virtual address in EBX
                             <1>       ; EAX = memory page address
000052FD 0C07                <1>       or     al, PTE_A_PRESENT + PTE_A_USER + PTE_A_WRITE
                             <1>                             ; present flag, bit 0 = 1
                             <1>                             ; user flag, bit 2 = 1
                             <1>                             ; writable flag, bit 1 = 1
000052FF 8902                <1>       mov    [edx], eax  ; Update PTE value
                             <1> gpa_6:
                             <1>       ; 18/10/2015
00005301 89D9                <1>       mov    ecx, ebx
00005303 81E1FF0F0000        <1>       and    ecx, PAGE_OFF
00005309 89C2                <1>       mov    edx, eax
0000530B 662500F0            <1>       and    ax, PTE_A_CLEAR
0000530F 01C8                <1>       add    eax, ecx
00005311 F7D9                <1>       neg    ecx ; 1 -> -1 (0FFFFFFFFh), 4095 (0FFFh) -> -4095
00005313 81C100100000        <1>       add    ecx, PAGE_SIZE
00005319 F8                  <1>       clc
                             <1> gpa_retn:
0000531A C3                  <1>       retn
                             <1> gpa_7:
0000531B E85AF8FFFF          <1>       call   allocate_page
00005320 72CE                <1>       jc     short gpa_im_err ; 'insufficient memory' error
00005322 E8CDF8FFFF          <1>       call   clear_page
00005327 EBA7                <1>       jmp    short gpa_2
                             <1>
                             <1> gpa_8: ; 26/03/2017
00005329 C605[0E6B0100]00    <1>       mov    byte [no_page_swap], 0
00005330 53                  <1>       push   ebx
00005331 50                  <1>       push   eax ; 26/03/2017
00005332 6681E300F0          <1>         and      bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
00005337 8A1D[B3030300]      <1>       mov    bl, [u.uno] ; current process number
0000533D E89DFDFFFF          <1>       call   swap_queue_shift ; drop from the queue if
                             <1>                            ; it is already on the queue
00005342 58                  <1>       pop    eax ; 26/03/2017
00005343 5B                  <1>       pop    ebx
00005344 C3                  <1>       retn
                             <1>
                             <1> reload_page:
                             <1>       ; 20/07/2015
                             <1>       ; 28/04/2015 (Retro UNIX 386 v1 - beginning)
                             <1>       ;
                             <1>       ; Reload (Restore) swapped page at memory
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      EBP = Virtual (linear) memory address
                             <1>       ;      EAX = PTE value (swap disk sector address)
                             <1>       ;      (Swap disk sector address = bit 1 to bit 31 of EAX)
                             <1>       ; OUTPUT ->
                             <1>       ;      EAX = PHYSICAL (real/flat) ADDRESS OF RELOADED PAGE
                             <1>       ;
                             <1>       ;      CF = 1 and EAX = error code
                             <1>       ;
                             <1>       ; Modified Registers -> none (except EAX)
                             <1>       ;
00005345 D1E8                <1>       shr    eax, 1   ; Convert PTE value to swap disk address
00005347 53                  <1>       push   ebx      ;
00005348 89C3                <1>       mov    ebx, eax ; Swap disk (offset) address
0000534A E82BF8FFFF          <1>       call   allocate_page
0000534F 720C                <1>       jc     short rlp_im_err
00005351 93                  <1>       xchg   eax, ebx
                             <1>       ; EBX = Physical memory (page) address
                             <1>       ; EAX = Swap disk (offset) address
                             <1>       ; EBP = Virtual (linear) memory address
00005352 E862FCFFFF          <1>       call   swap_in
00005357 720B                <1>       jc     short rlp_swp_err  ; (swap disk/file read error)
00005359 89D8                <1>       mov    eax, ebx
                             <1> rlp_retn:
0000535B 5B                  <1>       pop    ebx
```

```
0000535C C3                   <1>       retn
                              <1> 
                              <1> rlp_im_err:
0000535D B804000000           <1>       mov    eax, ERR_MINOR_IM ; Insufficient memory (minor) error!
                              <1>                             ; Major error = 0 (No protection fault)
00005362 EBF7                 <1>       jmp    short rlp_retn
                              <1> 
                              <1> rlp_swp_err:
00005364 B828000000           <1>       mov    eax, SWP_DISK_READ_ERR ; Swap disk read error !
00005369 EBF0                 <1>       jmp    short rlp_retn
                              <1> 
                              <1> 
                              <1> copy_page_dir:
                              <1>       ; 19/09/2015
                              <1>       ; temporary - 07/09/2015
                              <1>       ; 07/09/2015 (Retro UNIX 386 v1 - beginning)
                              <1>       ;
                              <1>       ; INPUT ->
                              <1>       ;      [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's
                              <1>       ;                page directory.
                              <1>       ; OUTPUT ->
                              <1>       ;      EAX =  PHYSICAL (real/flat) ADDRESS of the child's
                              <1>       ;             page directory.
                              <1>       ;      (New page directory with new page table entries.)
                              <1>       ;      (New page tables with read only copies of the parent's
                              <1>       ;      pages.)
                              <1>       ;      EAX = 0 -> Error (CF = 1)
                              <1>       ;
                              <1>       ; Modified Registers -> none (except EAX)
                              <1>       ;
0000536B E80AF8FFFF           <1>       call   allocate_page
00005370 723E                 <1>       jc     short cpd_err
                              <1>       ;
00005372 55                   <1>       push   ebp ; 20/07/2015
00005373 56                   <1>       push   esi
00005374 57                   <1>       push   edi
00005375 53                   <1>       push   ebx
00005376 51                   <1>       push   ecx
00005377 8B35[B8030300]       <1>       mov    esi, [u.pgdir]
0000537D 89C7                 <1>       mov    edi, eax
0000537F 50                   <1>       push   eax ; save child's page directory address
                              <1>       ; copy PDE 0 from the parent's page dir to the child's page dir
                              <1>       ; (use same system space for all user page tables)
00005380 A5                   <1>       movsd
00005381 BD00004000           <1>       mov    ebp, 1024*4096 ; pass the 1st 4MB (system space)
00005386 B9FF030000           <1>       mov    ecx, (PAGE_SIZE / 4) - 1 ; 1023
                              <1> cpd_0:
0000538B AD                   <1>       lodsd
                              <1>       ;or    eax, eax
                              <1>         ;jnz     short cpd_1
0000538C A801                 <1>       test   al, PDE_A_PRESENT ;  bit 0 =  1
0000538E 7508                 <1>       jnz    short cpd_1
                              <1>       ; (virtual address at the end of the page table)
00005390 81C500004000         <1>       add    ebp, 1024*4096 ; page size * PTE count
00005396 EB0F                 <1>       jmp    short cpd_2
                              <1> cpd_1:
00005398 662500F0             <1>       and    ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
0000539C 89C3                 <1>       mov    ebx, eax
                              <1>       ; EBX = Parent's page table address
0000539E E81F000000           <1>       call   copy_page_table
000053A3 720C                 <1>       jc     short cpd_p_err
                              <1>       ; EAX = Child's page table address
000053A5 0C07                 <1>       or     al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
                              <1>                     ; set bit 0, bit 1 and bit 2 to 1
                              <1>                     ; (present, writable, user)
                              <1> cpd_2:
000053A7 AB                   <1>       stosd
000053A8 E2E1                 <1>       loop   cpd_0
                              <1>       ;
000053AA 58                   <1>       pop    eax  ; restore child's page directory address
                              <1> cpd_3:
000053AB 59                   <1>       pop    ecx
000053AC 5B                   <1>       pop    ebx
000053AD 5F                   <1>       pop    edi
000053AE 5E                   <1>       pop    esi
000053AF 5D                   <1>       pop    ebp
                              <1> cpd_err:
000053B0 C3                   <1>       retn
                              <1> cpd_p_err:
                              <1>       ; release the allocated pages missing (recover free space)
000053B1 58                   <1>       pop    eax  ; the new page directory address (physical)
000053B2 8B1D[B8030300]       <1>       mov    ebx, [u.pgdir] ; parent's page directory address
000053B8 E8F6F8FFFF           <1>       call   deallocate_page_dir
000053BD 29C0                 <1>       sub    eax, eax ; 0
000053BF F9                   <1>       stc
000053C0 EBE9                 <1>       jmp    short cpd_3
                              <1> 
                              <1> copy_page_table:
                              <1>       ; 19/09/2015
                              <1>       ; temporary - 07/09/2015
                              <1>       ; 07/09/2015 (Retro UNIX 386 v1 - beginning)
                              <1>       ;
                              <1>       ; INPUT ->
                              <1>       ;      EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.
                              <1>       ;      EBP = page table entry index (from 'copy_page_dir')
                              <1>       ; OUTPUT ->
                              <1>       ;      EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.
                              <1>       ;      EBP = (recent) page table index (for 'add_to_swap_queue')
                              <1>       ;      CF = 1 -> error
                              <1>       ;
                              <1>       ; Modified Registers -> EBP (except EAX)
                              <1>       ;
000053C2 E8B3F7FFFF           <1>       call   allocate_page
```

```
000053C7 725A                 <1>       jc     short cpt_err
                              <1>       ;
000053C9 50                   <1>       push   eax ; *
                              <1>       ;push  ebx
000053CA 56                   <1>       push   esi
000053CB 57                   <1>       push   edi
000053CC 52                   <1>       push   edx
000053CD 51                   <1>       push   ecx
                              <1>       ;
000053CE 89DE                 <1>       mov    esi, ebx
000053D0 89C7                 <1>       mov    edi, eax
000053D2 89C2                 <1>       mov    edx, eax
000053D4 81C200100000         <1>       add    edx, PAGE_SIZE
                              <1> cpt_0:
000053DA AD                   <1>       lodsd
000053DB A801                 <1>       test   al, PTE_A_PRESENT ;  bit 0 =  1
000053DD 750B                 <1>       jnz    short cpt_1
000053DF 21C0                 <1>       and    eax, eax
000053E1 7430                 <1>       jz     short cpt_2
                              <1>       ; ebp = virtual (linear) address of the memory page
000053E3 E85DFFFFFF           <1>       call   reload_page ; 28/04/2015
000053E8 7234                 <1>       jc     short cpt_p_err
                              <1> cpt_1:
000053EA 662500F0             <1>       and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
000053EE 89C1                 <1>       mov    ecx, eax
                              <1>       ; Allocate a new page for the child process
000053F0 E885F7FFFF           <1>       call   allocate_page
000053F5 7227                 <1>       jc     short cpt_p_err
000053F7 57                   <1>       push   edi
000053F8 56                   <1>       push   esi
000053F9 89CE                 <1>       mov    esi, ecx
000053FB 89C7                 <1>       mov    edi, eax
000053FD B900040000           <1>       mov    ecx, PAGE_SIZE/4
00005402 F3A5                 <1>       rep    movsd  ; copy page (4096 bytes)
00005404 5E                   <1>       pop    esi
00005405 5F                   <1>       pop    edi
                              <1>       ;
00005406 53                   <1>       push   ebx
00005407 50                   <1>       push   eax
00005408 89EB                 <1>       mov    ebx, ebp
                              <1>       ; ebx = virtual address of the memory page
0000540A E8AAFDFFFF           <1>       call   add_to_swap_queue
0000540F 58                   <1>       pop    eax
00005410 5B                   <1>       pop    ebx
                              <1>       ;
                              <1>       ;or    ax, PTE_A_USER+PTE_A_PRESENT
00005411 0C07                 <1>       or     al, PTE_A_USER+PTE_A_WRITE+PTE_A_PRESENT
                              <1> cpt_2:
00005413 AB                   <1>       stosd  ; EDI points to child's PTE
                              <1>       ;
00005414 81C500100000         <1>       add    ebp, 4096 ; 20/07/2015 (next page)
                              <1>       ;
0000541A 39D7                 <1>       cmp    edi, edx
0000541C 72BC                 <1>       jb     short cpt_0
                              <1> cpt_p_err:
0000541E 59                   <1>       pop    ecx
0000541F 5A                   <1>       pop    edx
00005420 5F                   <1>       pop    edi
00005421 5E                   <1>       pop    esi
                              <1>       ;pop   ebx
00005422 58                   <1>       pop    eax ; *
                              <1> cpt_err:
00005423 C3                   <1>       retn
                              <1>
                              <1> allocate_memory_block:
                              <1>       ; 01/05/2017
                              <1>       ; 28/04/2017
                              <1>       ; 25/04/2017
                              <1>       ; 01/04/2016, 02/04/2016, 03/04/2016
                              <1>       ; 13/03/2016, 14/03/2016
                              <1>       ; 12/03/2016 (TRDOS 386 = TRDOS v2.0)
                              <1>       ; Allocating contiguous memory pages (in the kernel's memory space)
                              <1>       ;
                              <1>       ; INPUT ->
                              <1>       ;      EAX = Beginning address (physical)
                              <1>       ;      EAX = 0 -> Allocate memory block from the first proper aperture
                              <1>       ;      ECX = Number of bytes to be allocated
                              <1>       ;
                              <1>       ; OUTPUT ->
                              <1>       ;      1) cf = 0 -> successful
                              <1>       ;      EAX = Beginning (physical) address of the allocated memory block
                              <1>       ;      ECX = Number of allocated bytes (rounded up to page borders)
                              <1>       ;      2) cf = 1 -> unsuccessful
                              <1>       ;       2.1) If EAX > 0 ->
                              <1>       ;            (Number of requested pages is more than # of free pages
                              <1>       ;             but contiguous free pages -the aperture- is not enough!)
                              <1>       ;            EAX = Beginning address of available aperture
                              <1>       ;                 (one of all aperture with max. aperture size/length)
                              <1>       ;            ECX = Size of available aperture (memory block) in bytes
                              <1>       ;       2.2) If EAX = 0 -> Out of memory error
                              <1>       ;                  (number of free pages is less than requested number)
                              <1>       ;            ECX = Total number of free bytes (free pages * 4096)
                              <1>       ;                 (It is not number of contiguous free bytes)
                              <1>       ;
                              <1>       ; (Modified Registers -> EAX, ECX)
                              <1>       ;
                              <1>       ; PURPOSE: Loading a file at memory for copying or running etc.
                              <1>       ; If this procedure returns with cf is set, ECX contains maximum
                              <1>       ; available space and EAX contains the beginning address of it.
                              <1>       ; If EAX has zero, ECX contains total number of free bytes.
                              <1>       ; If requested block has been successfully allocated (by rounding up to
                              <1>       ; the last page border), it must be deallocated later by using
                              <1>       ; 'deallocate_memory_block' procedure.
```

```
                             <1>
00005424 52                  <1>       push   edx ; *
00005425 BAFF0F0000          <1>       mov    edx, PAGE_SIZE - 1   ; 4095
0000542A 01D0                <1>       add    eax, edx
0000542C 01D1                <1>       add    ecx, edx
0000542E C1E90C              <1>       shr    ecx, PAGE_SHIFT          ; 12
                             <1>
                             <1>       ; ECX = number of contiguous pages to be allocated
00005431 8B15[40580100]      <1>       mov    edx, [free_pages]
                             <1>       ; 01/05/2017
                             <1>       ;or    ecx, ecx
                             <1>       ;jz    short amb3
                             <1>       ; If ECX=0, set cf to 1 and return with max. available mem block size
                             <1>
00005437 39D1                <1>       cmp    ecx, edx
00005439 7760                <1>       ja     short amb_3
                             <1>
0000543B C1E80C              <1>       shr    eax, PAGE_SHIFT      ; 12
                             <1>
0000543E 89C2                <1>       mov    edx, eax          ; page number
00005440 C1EA03              <1>       shr    edx, 3            ; to get offset to M.A.T.
                             <1>                                ; (1 allocation bit = 1 page)
                             <1>                                ; (1 allocation bytes = 8 pages)
00005443 80E2FC              <1>       and    dl, 0FCh          ; clear lower 2 bits
                             <1>                                ; (to get 32 bit position)
00005446 53                  <1>       push   ebx ; **
                             <1> amb_0:
00005447 890D[F8640100]      <1>       mov    [mem_ipg_count], ecx ; initial (reset) value of page count
0000544D 890D[FC640100]      <1>       mov    [mem_pg_count], ecx
00005453 31C9                <1>       xor    ecx, ecx ; 0
00005455 890D[00650100]      <1>       mov    [mem_aperture], ecx ; 0
0000545B 890D[04650100]      <1>       mov    [mem_max_aperture], ecx ; 0
                             <1>
00005461 BB00001000          <1>       mov    ebx, MEM_ALLOC_TBL   ; Memory Allocation Table address.
00005466 3B15[44580100]      <1>       cmp    edx, [next_page]     ; Is the beginning page address lower
                             <1>                                ; than the address in 'next_page' ?
                             <1>                                ; (the first/next free page of user space)
0000546C 7208                <1>       jb     short amb_1
0000546E 3B15[48580100]      <1>       cmp    edx, [last_page]     ; is the beginning page address higher
                             <1>                                ; than the address in 'last_page' ?
                             <1>                                ; (end of the memory)
00005474 7606                <1>       jna    short amb_2       ; no
                             <1> amb_1:
00005476 8B15[44580100]      <1>       mov    edx, [next_page]     ; M.A.T. offset (1 M.A.T. byte = 8 pages)
                             <1> amb_2:
0000547C 01D3                <1>       add    ebx, edx
                             <1>
                             <1>       ; 28/04/2017
                             <1>       ;xor   ecx, ecx
0000547E 0FBC0B              <1>       bsf    ecx, [ebx]        ; 0 to 31
00005481 89D0                <1>       mov    eax, edx
00005483 C1E003              <1>       shl    eax, 3            ; *8
00005486 01C8                <1>       add    eax, ecx          ; beginning page number
                             <1>
00005488 A3[08650100]        <1>       mov    [mem_pg_pos], eax    ; beginning page no (for curr. mem. aperture)
0000548D A3[0C650100]        <1>       mov    [mem_max_pg_pos], eax ; beginning page no for max. mem. aperture
                             <1>
00005492 83E01F              <1>       and    eax, 1Fh          ; lower 5 bits only (0 to 31)
                             <1>                                ; (allocation bit position)
00005495 750E                <1>       jnz    short amb_4       ; 0
00005497 B120                <1>       mov    cl, 32
00005499 EB4B                <1>       jmp    short amb_10
                             <1>
                             <1> amb_3:      ; out_of_memory
0000549B 31C0                <1>       xor    eax, eax ; 0
0000549D 89D1                <1>       mov    ecx, edx ; free pages
0000549F C1E10C              <1>       shl    ecx, PAGE_SHIFT
000054A2 5A                  <1>       pop    edx ; *
000054A3 F9                  <1>       stc
000054A4 C3                  <1>       retn
                             <1> amb_4:
000054A5 8B13                <1>       mov    edx, [ebx]
000054A7 88C1                <1>       mov    cl, al ; 1 to 31
000054A9 D3EA                <1>       shr    edx, cl
000054AB 89D0                <1>       mov    eax, edx
                             <1> amb_5:
000054AD D1E8                <1>       shr    eax, 1 ; (***)
000054AF 7317                <1>       jnc    short amb_7
000054B1 FF05[00650100]      <1>       inc    dword [mem_aperture]
000054B7 FF0D[FC640100]      <1>       dec    dword [mem_pg_count]
000054BD 7470                <1>       jz     short amb_15
                             <1> amb_6:
                             <1>       ; 28/04/2017
000054BF FEC1                <1>       inc    cl
000054C1 80F920              <1>       cmp    cl, 32
000054C4 730D                <1>       jnb    short amb_9
000054C6 EBE5                <1>       jmp    short amb_5
                             <1> amb_7:
000054C8 50                  <1>       push   eax ; (***) allocation bits (in shifted status)
000054C9 E81B010000          <1>       call   amb_26 ; set maximum memory aperture (free memory block size)
000054CE 58                  <1>       pop    eax ; (***)
000054CF EBEE                <1>       jmp    short amb_6
                             <1> amb_8:
                             <1>       ; 28/04/2017
000054D1 B120                <1>       mov    cl, 32
                             <1> amb_9:
000054D3 89DA                <1>       mov    edx, ebx
000054D5 81EA00001000        <1>       sub    edx, MEM_ALLOC_TBL
000054DB 3B15[48580100]      <1>       cmp    edx, [last_page]
000054E1 7336                <1>       jnb    short amb_14 ; contiguous pages not enough
000054E3 83C304              <1>       add    ebx, 4
                             <1> amb_10:
000054E6 8B03                <1>       mov    eax, [ebx]
```

```
000054E8 21C0                <1>       and    eax, eax
000054EA 7408                <1>         jz      short amb_11 ; there is not a free page bit in this alloc dword
000054EC 40                  <1>       inc    eax ; 0FFFFFFFFh -> 0
000054ED 740C                <1>       jz     short amb_12 ; all of bits are set (32 free pages)
000054EF 48                  <1>       dec    eax
000054F0 28C9                <1>       sub    cl, cl ; 0
000054F2 EBB9                <1>       jmp    short amb_5
                             <1> amb_11:
000054F4 E8F0000000          <1>       call   amb_26 ; set maximum memory aperture (free memory block size)
000054F9 EBD8                <1>       jmp    short amb_9
                             <1> amb_12:
000054FB 390D[FC640100]      <1>       cmp    [mem_pg_count], ecx ; 32
00005501 7306                <1>       jnb    short amb_13
00005503 8B0D[FC640100]      <1>       mov    ecx, [mem_pg_count]
                             <1> amb_13:
00005509 010D[00650100]      <1>       add    [mem_aperture], ecx
0000550F 290D[FC640100]      <1>       sub    [mem_pg_count], ecx
00005515 7618                <1>       jna    short amb_15
00005517 EBBA                <1>       jmp    short amb_9 ; 01/05/2017
                             <1> amb_14:
00005519 E8CB000000          <1>       call   amb_26 ; 28/04/2017
0000551E A1[0C650100]        <1>       mov    eax, [mem_max_pg_pos] ; begin address of max. mem aperture
00005523 8B0D[04650100]      <1>       mov    ecx, [mem_max_aperture] ; max. (largest) memory aperture
00005529 F9                  <1>       stc
0000552A E9AF000000          <1>         jmp     amb_25
                             <1>
                             <1> amb_15: ; OK !
0000552F A1[08650100]        <1>       mov    eax, [mem_pg_pos]    ; Beginning address as page number
00005534 8B0D[00650100]      <1>       mov    ecx, [mem_aperture]  ; Free contiguous page count (>=1)
                             <1> amb_16:
                             <1>       ; allocate contiguous memory pages (via memory allocation table bits)
0000553A 89C2                <1>       mov    edx, eax
                             <1>       ; 25/04/2017
0000553C C1EA03              <1>       shr    edx, 3        ; 8 pages in one allocation byte
0000553F 80E2FC              <1>       and    dl, 0FCh      ; clear lower 2 bits
                             <1>                            ; (for dword/32bit positioning)
                             <1>
00005542 BB00001000          <1>       mov    ebx, MEM_ALLOC_TBL
00005547 01D3                <1>       add    ebx, edx
00005549 83E01F              <1>       and    eax, 1Fh ; 31
                             <1>       ; 03/04/2016
0000554C BA20000000          <1>       mov    edx, 32
00005551 28C2                <1>       sub    dl, al
00005553 39CA                <1>       cmp    edx, ecx      ; ecx >= 1
00005555 7602                <1>       jna    short amb_17
00005557 89CA                <1>       mov    edx, ecx
                             <1> amb_17:
00005559 29D1                <1>       sub    ecx, edx
0000555B 51                  <1>       push   ecx ; ***
0000555C 89D1                <1>       mov    ecx, edx
                             <1> amb_18:
0000555E 0FB303              <1>       btr    [ebx], eax    ; The destination bit indexed by the source value
                             <1>                            ; is copied into the Carry Flag and then cleared
                             <1>                            ; in the destination.
00005561 FF0D[40580100]      <1>       dec     dword [free_pages] ; 1 page has been allocated (X = X-1)
00005567 49                  <1>       dec    ecx
00005568 7404                <1>       jz     short amb_19
0000556A FEC0                <1>       inc    al
0000556C EBF0                <1>       jmp    short amb_18
                             <1> amb_19:
0000556E 59                  <1>       pop    ecx ; ***
0000556F 21C9                <1>       and    ecx, ecx ; 0 ?
00005571 741E                <1>       jz     short amb_22
                             <1>       ; 01/04/2016
00005573 B020                <1>       mov    al, 32
                             <1> amb_20:
00005575 83C304              <1>       add    ebx, 4
00005578 39C1                <1>       cmp    ecx, eax ; 32
0000557A 7305                <1>       jnb    short amb_21
                             <1>       ; ECX < 32
0000557C 28C0                <1>       sub    al, al ; 0
0000557E 50                  <1>       push   eax ; 0 ***
0000557F EBDD                <1>       jmp    short amb_18
                             <1> amb_21:
00005581 2905[40580100]      <1>       sub    [free_pages], eax   ; [free_pages] = [free_pages] - 32
00005587 C70300000000        <1>       mov    dword [ebx], 0          ; reset 32 bits
0000558D 29C1                <1>       sub    ecx, eax ; 32
0000558F 75E4                <1>       jnz    short amb_20
                             <1> amb_22:
00005591 A1[08650100]        <1>       mov    eax, [mem_pg_pos]   ; Beginning address as page number
00005596 8B0D[00650100]      <1>       mov    ecx, [mem_aperture] ; Free contiguous page count
                             <1>       ; [next_page] update
0000559C 89C2                <1>       mov    edx, eax
                             <1>       ; 03/04/2016
0000559E C1EA03              <1>       shr    edx, 3            ; to get offset to M.A.T.
                             <1>                                ; (1 allocation bit = 1 page)
                             <1>                                ; (1 allocation bytes = 8 pages)
000055A1 80E2FC              <1>       and    dl, 0FCh          ; clear lower 2 bits
                             <1>                                ; (to get 32 bit position)
000055A4 3B15[44580100]      <1>       cmp    edx, [next_page] ; first free page pointer offset
000055AA 7732                <1>       ja     short amb_25
000055AC BB00001000          <1>       mov    ebx, MEM_ALLOC_TBL
000055B1 833C1300            <1>       cmp    dword [ebx+edx], 0
000055B5 7721                <1>       ja     short amb_24
000055B7 89C2                <1>       mov    edx, eax
000055B9 01CA                <1>       add    edx, ecx
000055BB C1EA03              <1>       shr    edx, 3
000055BE 80E2FC              <1>       and    dl, 0FCh
                             <1> amb_23:
000055C1 833C1300            <1>       cmp    dword [ebx+edx], 0
000055C5 7711                <1>       ja     short amb_24
000055C7 83C204              <1>       add    edx, 4
000055CA 3B15[48580100]      <1>       cmp    edx, [last_page]    ; last page pointer offset
```

```
000055D0 76EF                <1>       jna    short amb_23
000055D2 8B15[4C580100]      <1>       mov    edx, [first_page]   ; (for) beginning of user's space
                             <1> amb_24:
000055D8 8915[44580100]      <1>       mov    [next_page], edx
                             <1> amb_25:
000055DE 9C                  <1>       pushf
000055DF C1E00C              <1>       shl    eax, PAGE_SHIFT         ; convert to phy. address in bytes
000055E2 C1E10C              <1>       shl    ecx, PAGE_SHIFT         ; convert to byte counts
000055E5 9D                  <1>       popf
000055E6 5B                  <1>       pop    ebx ; **
000055E7 5A                  <1>       pop    edx ; *
000055E8 C3                  <1>       retn
                             <1>
                             <1> amb_26:     ; set maximum free memory aperture (free memory block size)
000055E9 89DA                <1>       mov    edx, ebx ; current address
000055EB 81EA00001000        <1>       sub    edx, MEM_ALLOC_TBL ; MAT beginning address
                             <1>       ; 02/04/2016
000055F1 C1E203              <1>       shl    edx, 3 ; MAT byte offset * 8 = page number base
000055F4 01CA                <1>       add    edx, ecx ; current page number (ecx = 0 to 32)
                             <1>       ;
000055F6 A1[00650100]        <1>       mov    eax, [mem_aperture]
000055FB 21C0                <1>       and    eax, eax
000055FD 7421                <1>         jz      short amb_27
000055FF C705[00650100]0000- <1>         mov     dword [mem_aperture], 0
00005607 0000                <1>
00005609 3B05[04650100]      <1>       cmp    eax, [mem_max_aperture]
0000560F 760F                <1>       jna    short amb_27
00005611 A3[04650100]        <1>       mov    [mem_max_aperture], eax
                             <1>       ; 25/04/2017
00005616 A1[08650100]        <1>       mov    eax, [mem_pg_pos]
                             <1>       ; EAX = Beginning page number of the max. aperture
0000561B A3[0C650100]        <1>       mov    [mem_max_pg_pos], eax
                             <1> amb_27:
00005620 8915[08650100]      <1>       mov    [mem_pg_pos], edx ; current page
                             <1>
00005626 A1[F8640100]        <1>       mov    eax, [mem_ipg_count] ; initial (reset) value of page count
0000562B A3[FC640100]        <1>       mov    [mem_pg_count], eax
                             <1>
00005630 C3                  <1>       retn
                             <1>
                             <1> deallocate_memory_block:
                             <1>       ; 03/04/2016
                             <1>       ; 14/03/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; Deallocating contiguous memory pages (in the kernel's memory space)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      EAX = Beginning address (physical)
                             <1>       ;      ECX = Number of bytes to be deallocated
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      Memory Allocation Table bits will be updated
                             <1>       ;      [free_pages] will be changed (increased)
                             <1>       ;
                             <1>       ; (Modified Registers -> EAX, ECX)
                             <1>       ;
                             <1>       ; PURPOSE: Unloading/Freeing a file -or an allocated memory block-
                             <1>       ; at memory after copying, running, saving, reading, writing etc.
                             <1>       ;
                             <1>
00005631 52                  <1>       push   edx ; *
00005632 53                  <1>       push   ebx ; **
                             <1>
00005633 C1E80C              <1>       shr    eax, PAGE_SHIFT         ; 12
00005636 C1E90C              <1>       shr    ecx, PAGE_SHIFT         ; 12
                             <1>
                             <1>       ; EAX = Beginning page number
                             <1>       ; ECX = Number of contiguous pages to be deallocated
                             <1> damb_0:
                             <1>       ; deallocate contiguous memory pages (via memory allocation table bits)
00005639 89C2                <1>       mov    edx, eax
0000563B C1EA03              <1>       shr    edx, 3            ; to get offset to M.A.T.
                             <1>                                ; (1 allocation bit = 1 page)
                             <1>                                ; (1 allocation bytes = 8 pages)
0000563E 80E2FC              <1>       and    dl, 0FCh          ; clear lower 2 bits
                             <1>                                ; (to get 32 bit position)
00005641 3B15[44580100]      <1>       cmp    edx, [next_page] ; next free page
00005647 7306                <1>       jnb    short damb_1
00005649 8915[44580100]      <1>       mov    [next_page], edx
                             <1> damb_1:
0000564F BB00001000          <1>       mov    ebx, MEM_ALLOC_TBL
00005654 01D3                <1>       add    ebx, edx
00005656 83E01F              <1>       and    eax, 1Fh ; 31
                             <1>
                             <1>       ; 03/04/2016
00005659 BA20000000          <1>       mov    edx, 32
0000565E 28C2                <1>       sub    dl, al
00005660 39CA                <1>       cmp    edx, ecx
00005662 7602                <1>       jna    short damb_2
00005664 89CA                <1>       mov    edx, ecx
                             <1> damb_2:
00005666 29D1                <1>       sub    ecx, edx
00005668 51                  <1>       push   ecx ; ***
00005669 89D1                <1>       mov    ecx, edx
                             <1> damb_3:
0000566B 0FAB03              <1>       bts    [ebx], eax        ; unlink/release/deallocate page
                             <1>                                ; set relevant bit to 1.
                             <1>                                ; set CF to the previous bit value
0000566E FF05[40580100]      <1>       inc     dword [free_pages]   ; 1 page has been deallocated (X = X+1)
00005674 49                  <1>       dec    ecx
00005675 7404                <1>       jz     short damb_4
00005677 FEC0                <1>       inc    al
00005679 EBF0                <1>       jmp    short damb_3
                             <1> damb_4:
```

```
0000567B 59                  <1>       pop    ecx ; ***
0000567C 21C9                <1>       and    ecx, ecx ; 0 ?
0000567E 741E                <1>       jz     short damb_7
                             <1>       ; 03/04/2016
00005680 B020                <1>       mov    al, 32
                             <1> damb_5:
00005682 83C304              <1>       add    ebx, 4
00005685 39C1                <1>       cmp    ecx, eax ; 32
00005687 7305                <1>       jnb    short damb_6
                             <1>       ; ECX < 32
00005689 28C0                <1>       sub    al, al ; 0
0000568B 50                  <1>       push   eax ; 0 ***
0000568C EBDD                <1>       jmp    short damb_3
                             <1> damb_6:
0000568E 0105[40580100]      <1>       add    [free_pages], eax ; [free_pages] = [free_pages] + 32
00005694 C703FFFFFFFF        <1>       mov    dword [ebx], 0FFFFFFFFh ; set 32 bits
0000569A 29C1                <1>       sub    ecx, eax ; 32
0000569C 75E4                <1>       jnz    short damb_5
                             <1> damb_7:
0000569E 5B                  <1>       pop    ebx ; **
0000569F 5A                  <1>       pop    edx ; *
000056A0 C3                  <1>       retn
                             <1>
                             <1> direct_memory_access:
                             <1>       ; 22/07/2017
                             <1>       ; 12/05/2017
                             <1>       ; 16/07/2016
                             <1>       ; 12/07/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; This processure will be called to map
                             <1>       ; user's (ring 3) page tables to access phsical
                             <1>       ; (flat/linear) memory addresses, directly (without
                             <1>       ;  kernel's data transfer functions).
                             <1>       ;
                             <1>       ; Purpose: Video memory access and shared memory access.
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      EAX = Beginning address (physical).
                             <1>       ;      EBX = User's buffer address ; 12/05/2017
                             <1>       ;      ECX = Number of contiguous pages to be mapped.
                             <1>       ; OUTPUT ->
                             <1>       ;      User's page directory and pages tables
                             <1>       ;      will be updated.
                             <1>       ;
                             <1>       ;      If an old page table entry has valid page address,
                             <1>       ;      that page will be deallocated just before PTE will
                             <1>       ;      be changed for direct (1 to 1) memory page access.
                             <1>       ;
                             <1>       ;      If old PTE value points to a swapped page,
                             <1>         ;       that page (block) will be unlinked on swap disk.
                             <1>       ;
                             <1>       ;      Newly allocated pages (except page tables) will not
                             <1>       ;      be applied to Memory Allocation Table.
                             <1>       ;      AVL bit 1 (PTE bit 10) of page table entry will be
                             <1>       ;      used to indicate shared (direct) memory page; then,
                             <1>       ;      this page will not be deallocated later during
                             <1>       ;      process termination. (Memory Allocation Table and
                             <1>       ;      free memory count will not be affected.
                             <1>       ;      (Except deallocating page table's itself.)
                             <1>       ;
                             <1>       ;       CF = 1 -> error (EAX = error code)
                             <1>       ;       CF = 0 -> success (EAX = beginning address)
                             <1>       ;
                             <1>       ;; (Modified Registers -> none)
                             <1>       ; Modified registers: ebp, edx, ecx, ebx, esi, edi
                             <1>       ;
                             <1>
                             <1>       ;push  ebp
                             <1>       ;push  ebx
                             <1>       ;push  ecx
                             <1>       ;push  edx
000056A1 662500F0            <1>       and    ax, PTE_A_CLEAR ; clear page offset
000056A5 50                  <1>       push   eax
                             <1>       ;and   ecx, ecx ; page count
                             <1>       ;jz    dmem_acc_7  ; 'insufficient memory' error
000056A6 89C5                <1>       mov    ebp, eax
000056A8 81C300004000        <1>       add    ebx, CORE ; 12/05/2017
                             <1> dmem_acc_0:
000056AE 891D[F86F0100]      <1>       mov    [base_addr], ebx ; 12/05/2017
000056B4 A1[B8030300]        <1>       mov    eax, [u.pgdir] ; page dir address (physical)
000056B9 E8D7F5FFFF          <1>       call   get_pte
                             <1>              ; EDX = Page table entry address (if CF=0)
                             <1>                ;       Page directory entry address (if CF=1)
                             <1>              ;       (Bit 0 value is 0 if PT is not present)
                             <1>              ; EAX = Page table entry value (page address)
                             <1>              ;      CF = 1 -> PDE not present or invalid ?
000056BE 7324                <1>       jnc    short dmem_acc_1
                             <1>       ;
000056C0 E8B5F4FFFF          <1>       call   allocate_page
000056C5 0F82AB000000        <1>         jc      dmem_acc_7  ; 'insufficient memory' error
                             <1>       ;
000056CB E824F5FFFF          <1>       call   clear_page
                             <1>       ; EAX = Physical (base) address of the allocated (new) page
000056D0 0C07                <1>       or     al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER ; 4+2+1 = 7
                             <1>                        ; lower 3 bits are used as U/S, R/W, P flags
                             <1>                        ; (user, writable, present page)
000056D2 8902                <1>       mov    [edx], eax ; Let's put the new page directory entry here !
000056D4 A1[B8030300]        <1>       mov    eax, [u.pgdir]
000056D9 E8B7F5FFFF          <1>       call   get_pte
000056DE 0F8292000000        <1>         jc      dmem_acc_7 ; 'insufficient memory' error
                             <1> dmem_acc_1:
                             <1>       ; EAX = PTE value, EDX = PTE address
000056E4 A801                <1>       test   al, PTE_A_PRESENT
000056E6 750D                <1>       jnz    short dmem_acc_2
```

```
000056E8 09C0                <1>       or     eax, eax
000056EA 7468                <1>       jz     short dmem_acc_6   ; Change PTE
000056EC D1E8                <1>       shr    eax, 1       ; swap disk block (8 sectors) address
                             <1>       ; unlink swap disk block
000056EE E80CFBFFFF          <1>       call   unlink_swap_block
000056F3 EB5F                <1>       jmp    short dmem_acc_6
                             <1> 
                             <1> dmem_acc_2:
000056F5 A802                <1>       test   al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
                             <1>                             ; (must be 1)
000056F7 7550                <1>       jnz    short dmem_acc_4
                             <1>       ; Read only -duplicated- page (belongs to a parent or a child)
000056F9 66A90002            <1>         test    ax, PTE_DUPLICATED ; Was this page duplicated
                             <1>                              ; as child's page ?
000056FD 7455                <1>       jz     short dmem_acc_5 ; Change PTE but don't deallocate the page!
                             <1> 
                             <1>       ;push  edi
                             <1>       ;push  esi
                             <1> 
000056FF 51                  <1>       push   ecx
                             <1>       ;push  ebx
00005700 8B1D[BC030300]      <1>       mov    ebx, [u.ppgdir] ; parent's page dir address (physical)
                             <1> 
                             <1>       ; check the parent's PTE value is read only & same page or not..
00005706 89EF                <1>       mov    edi, ebp
00005708 C1EF16              <1>       shr    edi, PAGE_D_SHIFT ; 22
                             <1>       ; EDI = page directory entry index (0-1023)
0000570B 89EE                <1>       mov    esi, ebp
0000570D C1EE0C              <1>       shr    esi, PAGE_SHIFT ; 12
00005710 81E6FF030000        <1>       and    esi, PTE_MASK
                             <1>       ; ESI = page table entry index (0-1023)
                             <1> 
00005716 66C1E702            <1>       shl    di, 2 ; * 4
0000571A 01FB                <1>       add    ebx, edi ; PDE offset (for the parent)
0000571C 8B0F                <1>       mov    ecx, [edi]
0000571E F6C101              <1>       test   cl, PDE_A_PRESENT ; present (valid) or not ?
00005721 7425                <1>       jz     short dmem_acc_3    ; parent process does not use this page
00005723 6681E100F0          <1>       and    cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
00005728 66C1E602            <1>       shl    si, 2 ; *4
0000572C 01CE                <1>       add    esi, ecx ; PTE offset (for the parent)
0000572E 8B1E                <1>       mov    ebx, [esi]
00005730 F6C301              <1>       test   bl, PTE_A_PRESENT ; present or not ?
00005733 7413                <1>       jz     short dmem_acc_3    ; parent process does not use this page
00005735 662500F0            <1>       and    ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
00005739 6681E300F0          <1>       and    bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
0000573E 39D8                <1>       cmp    eax, ebx     ; parent's and child's pages are same ?
00005740 7506                <1>       jne    short dmem_acc_3    ; not same page
                             <1>                           ; deallocate the child's page
00005742 800E02              <1>         or      byte [esi], PTE_A_WRITE ; convert to writable page (parent)
                             <1>       ;pop   ebx
00005745 59                  <1>       pop    ecx
00005746 EB0C                <1>       jmp    short dmem_acc_5
                             <1> dmem_acc_3:
                             <1>       ;pop   ebx
00005748 59                  <1>       pop    ecx
                             <1> dmem_acc_4:
00005749 66A90004            <1>       test   ax, PTE_SHARED ; shared or direct memory access indicator
0000574D 7505                <1>       jnz    short dmem_acc_5   ; AVL bit 1 = 1, do not deallocate this page!
                             <1>       ;
                             <1>       ;and   ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
0000574F E804F6FFFF          <1>       call   deallocate_page
                             <1> dmem_acc_5:
                             <1>       ;pop   esi
                             <1>       ;pop   edi
                             <1> dmem_acc_6:
00005754 89E8                <1>       mov    eax, ebp ; physical page (offset=0) address
                             <1>       ; EAX = memory page address
                             <1>       ; EDX = PTE entry address (physical)
00005756 660D0704            <1>       or     ax, PTE_A_PRESENT+PTE_A_USER+PTE_A_WRITE+PTE_SHARED
                             <1>                    ; present flag, bit 0 = 1
                             <1>                    ; user flag, bit 2 = 1
                             <1>                    ; writable flag, bit 1 = 1
                             <1>                    ; direct memory access flag, bit 10 = 1
                             <1>                    ; (This page must not be deallocated!)
0000575A 8902                <1>       mov    [edx], eax  ; Update PTE value
0000575C 49                  <1>       dec    ecx ; remain count of contiguous pages
0000575D 741E                <1>       jz     short dmem_acc_8
0000575F 81C500100000        <1>       add    ebp, PAGE_SIZE ; next physical page address
                             <1>       ; 22/07/2017
                             <1>       ;mov   eax, ebp
                             <1>       ; 12/05/2017
00005765 8B1D[F86F0100]      <1>       mov    ebx, [base_addr] ; linear address (virtual+CORE)
0000576B 81C300100000        <1>       add    ebx, PAGE_SIZE      ; next linear address
00005771 E938FFFFFF          <1>         jmp     dmem_acc_0
                             <1> dmem_acc_7:  ; ERROR !
00005776 C7042404000000      <1>       mov    dword [esp], ERR_MINOR_IM
                             <1>              ; Insufficient memory (minor) error!
                             <1>              ; Major error = 0 (No protection fault)
                             <1>       ; cf = 1
                             <1> dmem_acc_8:
0000577D 58                  <1>       pop    eax
                             <1>       ;pop   edx
                             <1>       ;pop   ecx
                             <1>       ;pop   ebx
                             <1>       ;pop   ebp
0000577E C3                  <1>       retn
                             <1> 
                             <1> deallocate_user_pages:
                             <1>       ; 20/05/2017
                             <1>       ; 15/05/2017
                             <1>       ; 20/02/2017
                             <1>       ; 19/02/2017 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>         ; Deallocate virtually contiguous user pages (memory block)
                             <1>         ; (caller: 'sysdalloc' system call)
                             <1>         ;
                             <1>         ; INPUT ->
                             <1>         ;      EBX = VIRTUAL ADDRESS (beginning address)
                             <1>         ;      ECX = byte count
                             <1>         ;      [u.pgdir] = user's page directory
                             <1>         ;      [u.ppdir] = parent's page directory
                             <1>         ;
                             <1>         ; OUTPUT ->
                             <1>         ;    If CF = 0
                             <1>         ;      EAX = Deallocated memory bytes
                             <1>         ;        (Even if shared or read only pages will not be
                             <1>         ;         deallocated on M.A.T., this byte count will be
                             <1>         ;         returned as virtually deallocated bytes; in fact
                             <1>         ;         virtually deallocated user pages * 4096.)
                             <1>         ;      EBX = Virtual address (as rounded up)
                             <1>         ;    If CF = 1
                             <1>         ;      EAX = 0 (there is not any deallocated pages)
                             <1>         ;
                             <1>         ; Note: Empty page tables will not be deallocated!!!
                             <1>         ;     (they will be deallocated at process termination stage)
                             <1>         ;
                             <1>         ; Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP
                             <1>         ;
0000577F 89DE                <1>         mov    esi, ebx
00005781 89F7                <1>         mov    edi, esi
00005783 01CF                <1>         add    edi, ecx
00005785 81C6FF0F0000        <1>         add    esi, PAGE_SIZE - 1  ; 4095 (round up)
0000578B C1EE0C              <1>         shr    esi, PAGE_SHIFT
0000578E C1EF0C              <1>         shr    edi, PAGE_SHIFT
00005791 89F8                <1>         mov    eax, edi ; end page
00005793 29F0                <1>         sub    eax, esi ; end page - start page
00005795 0F86D5000000        <1>         jna    da_u_pd_err  ; < 1
0000579B 89F3                <1>         mov    ebx, esi
0000579D C1E30C              <1>         shl    ebx, PAGE_SHIFT ; virtual address (as rounded up)
000057A0 53                  <1>         push   ebx ; *
000057A1 89C1                <1>         mov    ecx, eax ; page count
000057A3 C1E00C              <1>         shl    eax, PAGE_SHIFT ; byte count as adjusted
000057A6 50                  <1>         push   eax ; **
000057A7 8B1D[B8030300]      <1>         mov    ebx, [u.pgdir] ; physical addr of user's page dir
000057AD 81C600040000        <1>         add    esi, CORE/PAGE_SIZE
000057B3 89F7                <1>         mov    edi, esi
000057B5 81E7FF030000        <1>         and    edi, PTE_MASK ; PTE entry in the page table
000057BB 57                  <1>         push   edi ; *** ; PTE index (of page directory)
000057BC C1EE0A              <1>         shr    esi, PAGE_D_SHIFT - PAGE_SHIFT ; 22-12=10
000057BF 89F2                <1>         mov    edx, esi
                             <1>         ; EDX = PDE index
000057C1 C1E602              <1>         shl    esi, 2 ; convert PDE index to dword offset
000057C4 01DE                <1>         add    esi, ebx ; add page directory address
                             <1> da_u_pd_1:
000057C6 AD                  <1>         lodsd
                             <1>         ;
000057C7 89F5                <1>         mov    ebp, esi ; 20/02/2017
                             <1>         ; EBP = next PDE address
                             <1>         ;
000057C9 A801                <1>         test   al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
000057CB 0F8494000000        <1>         jz     da_u_pd_3 ; 20/05/2017
000057D1 662500F0            <1>         and    ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
                             <1>         ; EAX = PHYSICAL (flat) ADDRESS OF THE PAGE TABLE
000057D5 8B3C24              <1>         mov    edi, [esp] ; ***
                             <1>         ; EDI = PTE index (of complete page directory)
                             <1>         ;and   edi, PTE_MASK ; PTE entry in the page table
000057D8 C1E702              <1>         shl    edi, 2 ; convert PTE index to dword offset
000057DB 89FE                <1>         mov    esi, edi ; PTE offset in page table (0-4092)
000057DD 01C6                <1>         add    esi, eax ; now, esi points to requested PTE
                             <1> da_u_pt_0:
000057DF AD                  <1>         lodsd
000057E0 A801                <1>         test   al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
000057E2 743F                <1>         jz     short da_u_pt_1
                             <1>         ;
000057E4 A802                <1>         test   al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
                             <1>                               ; (must be 1)
000057E6 7549                <1>         jnz    short da_u_pt_3
                             <1>         ; Read only -duplicated- page (belongs to a parent or a child)
000057E8 66A90002            <1>           test    ax, PTE_DUPLICATED ; Was this page duplicated
                             <1>                                ; as child's page ?
000057EC 744E                <1>         jz     short da_u_pt_4 ; Clear PTE but don't deallocate the page!
                             <1>         ;
                             <1>         ; check the parent's PTE value is read only & same page or not..
                             <1>         ; EDX = page directory entry index (0-1023)
000057EE 52                  <1>         push   edx ; ****
                             <1>         ; EDI = page table entry offset (0-4092)
000057EF 8B1D[BC030300]      <1>         mov    ebx, [u.ppgdir] ; page directory of the parent process
000057F5 66C1E202            <1>         shl    dx, 2 ; *4
000057F9 01D3                <1>         add    ebx, edx ; PDE address (for the parent)
000057FB 8B13                <1>         mov    edx, [ebx] ; page table address
000057FD F6C201              <1>         test   dl, PDE_A_PRESENT ; present (valid) or not ?
00005800 742E                <1>         jz     short da_u_pt_2     ; parent process does not use this page
00005802 6681E200F0          <1>         and    dx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
                             <1>         ; EDI = page table entry offset (0-4092)
00005807 01D7                <1>         add    edi, edx      ; PTE address (for the parent)
00005809 8B1F                <1>         mov    ebx, [edi]
0000580B F6C301              <1>         test   bl, PTE_A_PRESENT ; present or not ?
0000580E 7420                <1>         jz     short da_u_pt_2     ; parent process does not use this page
00005810 662500F0            <1>         and    ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
00005814 6681E300F0          <1>         and    bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
00005819 39D8                <1>         cmp    eax, ebx      ; parent's and child's pages are same ?
0000581B 7513                <1>         jne    short da_u_pt_2     ; not same page
                             <1>                            ; deallocate the child's page
0000581D 800F02              <1>           or      byte [edi], PTE_A_WRITE ; convert to writable page (parent)
00005820 5A                  <1>         pop    edx ; ****
00005821 EB19                <1>         jmp    short da_u_pt_4
```

```
                             <1> da_u_pt_1:
00005823 09C0                <1>       or     eax, eax     ; swapped page ?
00005825 741C                <1>       jz     short da_u_pt_5     ; no
                             <1>                          ; yes
00005827 D1E8                <1>       shr    eax, 1
00005829 E8D1F9FFFF          <1>       call   unlink_swap_block ; Deallocate swapped page block
                             <1>                            ; on the swap disk (or in file)
0000582E EB13                <1>       jmp    short da_u_pt_5
                             <1> da_u_pt_2:
00005830 5A                  <1>       pop    edx ; ****
                             <1> da_u_pt_3:
00005831 66A90004            <1>       test   ax, PTE_SHARED      ; shared or direct memory access indicator
00005835 7505                <1>       jnz    short da_u_pt_4     ; AVL bit 1 = 1, do not deallocate this page!
                             <1>       ;
                             <1>       ;and   ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
00005837 E81CF5FFFF          <1>       call   deallocate_page ; set the mem allocation bit of this page
                             <1> da_u_pt_4:
0000583C C746FC00000000      <1>       mov    dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)
                             <1> da_u_pt_5:
                             <1>       ; 20/05/2017
00005843 58                  <1>       pop    eax ; *** PTE index (of page directory)
00005844 49                  <1>       dec    ecx ; remain page count
00005845 7426                <1>       jz     short da_u_pd_4
00005847 40                  <1>       inc    eax ; next PTE
00005848 6625FF03            <1>       and    ax, PTE_MASK ; PTE entry index in the page table
0000584C 50                  <1>       push   eax ; *** (save again)
                             <1>       ;mov   edi, eax
                             <1>       ;and   di, PTE_MASK
                             <1>       ;cmp   edi, PAGE_SIZE / 4 ; 1024
                             <1>       ;jnb   short da_u_pd_2
0000584D 89C7                <1>       mov    edi, eax
0000584F C1E702              <1>       shl    edi, 2 ; convert index to dword offset
                             <1>       ;test  ax, PTE_MASK ; 3FFh
00005852 09C0                <1>       or     eax, eax
00005854 7589                <1>       jnz    short da_u_pt_0 ; 1-1023
                             <1> da_u_pd_2:
00005856 42                  <1>       inc    edx
                             <1>       ; 20/05/2017
00005857 6681E2FF03          <1>       and    dx, PTE_MASK  ; 3FFh
0000585C 740F                <1>       jz     short da_u_pd_4  ; 0 (1024)
                             <1>       ;cmp   edx, 1024
                             <1>       ;jnb   short da_u_pd_4
0000585E 89EE                <1>       mov    esi, ebp ; 20/02/2017
00005860 E961FFFFFF          <1>       jmp    da_u_pd_1
                             <1> da_u_pd_3:
                             <1>       ; 15/05/2017 (empty page directory entry)
00005865 81E900040000        <1>       sub    ecx, 1024
0000586B 77E9                <1>       ja     short da_u_pd_2 ; 20/05/2017
                             <1> da_u_pd_4:
0000586D 58                  <1>       pop    eax ; **
0000586E 5B                  <1>       pop    ebx ; *
0000586F C3                  <1>       retn
                             <1>
                             <1> da_u_pd_err:
00005870 31C0                <1>       xor    eax, eax
00005872 F9                  <1>       stc
00005873 C3                  <1>       retn
                             <1>
                             <1> allocate_user_pages:
                             <1>       ; 20/05/2017
                             <1>       ; 01/05/2017, 02/05/2017, 15/05/2017
                             <1>       ; 04/03/2017
                             <1>       ; 20/02/2017 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>       ; Allocate physically contiguous user pages (memory block)
                             <1>       ; (caller: 'sysalloc' system call)
                             <1>       ;
                             <1>       ; Note: This procedure does not alloc a page's itself
                             <1>       ;      (page bit) on Memory Allocation Table.
                             <1>       ;      (allocate_memory_block is needed before this proc)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      EAX = PHYSICAL ADDRESS (beginning address)
                             <1>       ;      EBX = VIRTUAL ADDRESS (beginning address)
                             <1>       ;      ECX = byte count (>=4096)
                             <1>       ;      [u.pgdir] = user's page directory
                             <1>       ;
                             <1>       ;      Note: All addresses are (must be) already adjusted
                             <1>       ;      to page      borders, otherwise, lower 12bits of addresses
                             <1>       ;      and byte count would be truncated.
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      none
                             <1>       ;
                             <1>       ;      CF = 1 -> insufficient memory error
                             <1>       ;
                             <1>       ; Note: All pages will be allocated in physical page order
                             <1>       ;      from the beginning page address.
                             <1>       ;      * A new page table will be added to the page dir
                             <1>       ;        when the requested PDE is invalid.
                             <1>       ;      * Those pages will not be added to swap queue
                             <1>       ;        because main purpose of this allocation is to
                             <1>       ;        set a direct memory access (DMA controller) buffer.
                             <1>       ;       (Swapping out a page in a DMA buffer would be wrong!)
                             <1>       ;      * Previous content of page tables (PTEs) would be
                             <1>       ;        (should be) deallocated before entering this
                             <1>       ;        procedure. So, new page table entries (PTEs)
                             <1>       ;        directly will be written without checking
                             <1>       ;        their previous content.
                             <1>       ;      * Only solution to increase free memory by removing
                             <1>       ;        that non-swappable memory block is to terminate
                             <1>       ;        the process or to wait until the process will
                             <1>       ;        deallocate that memory block as itself. ('sysdalloc')
                             <1>        ;        (No problem, if the process does not grab all of
                             <1>        ;        -very big amount of- free memory by using
                             <1>        ;        'sysalloc' system call!?)
                             <1>        ;        (Even if the process has grabbed all of free memory,
                             <1>        ;        no problem if the process is not running in
                             <1>        ;        multitasking mode. No problem in multitasking
                             <1>        ;        mode if there is not another process which is running
                             <1>        ;        or waiting or sleeping for an event as it's pages
                             <1>        ;        are swapped-out. But a new process can not start to
                             <1>        ;        run if all of free memory has beeen allocated
                             <1>        ;        by running processes. Deallocation -'sysdalloc'-
                             <1>        ;        or terminate a running process is needed
                             <1>        ;        in order to run a new process.)
                             <1>        ;
                             <1>        ; Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP
                             <1>        ;
                             <1>
                             <1>        ; 01/05/2017
00005874 662500F0            <1>        and    ax, ~PAGE_OFF
00005878 6681E300F0          <1>        and    bx, ~PAGE_OFF
                             <1>        ; 02/05/2017
0000587D BD00F0FFFF          <1>        mov    ebp, 0FFFFF000h ; 4 Giga Bytes - 4096 Bytes (for Stack)
00005882 C1E90C              <1>        shr    ecx, PAGE_SHIFT ; page count
00005885 83F901              <1>        cmp    ecx, 1
00005888 7251                <1>        jb     short a_u_im_retn
0000588A 89C2                <1>        mov    edx, eax
0000588C 01CA                <1>        add    edx, ecx
0000588E 724B                <1>        jc     short a_u_im_retn
00005890 39D5                <1>        cmp    ebp, edx
00005892 7247                <1>        jb     short a_u_im_retn
00005894 89DA                <1>        mov    edx, ebx
00005896 81C200004000        <1>        add    edx, CORE
0000589C 723D                <1>        jc     short a_u_im_retn
0000589E 01CA                <1>        add    edx, ecx
000058A0 7239                <1>        jc     short a_u_im_retn
000058A2 39D5                <1>        cmp    ebp, edx
000058A4 7235                <1>        jb     short a_u_im_retn
                             <1>        ;
000058A6 89C5                <1>        mov    ebp, eax ; physical address
000058A8 89DE                <1>        mov    esi, ebx
000058AA 81C600004000        <1>        add    esi, CORE ; start of user's memory (4M)
000058B0 C1EE0C              <1>        shr    esi, PAGE_SHIFT ; higher 20 bits of the linear address
                             <1>        ;shr   ecx, PAGE_SHIFT ; page count
000058B3 8B1D[B8030300]      <1>        mov    ebx, [u.pgdir] ; physical addr of user's page dir
000058B9 89F7                <1>        mov    edi, esi
000058BB 81E7FF030000        <1>        and    edi, PTE_MASK ; PTE entry index in the page table
000058C1 57                  <1>        push   edi  ; * ; PTE index (in page directory)
000058C2 C1EE0A              <1>        shr    esi, PAGE_D_SHIFT - PAGE_SHIFT ; 22-12=10
000058C5 89F2                <1>        mov    edx, esi
                             <1>        ; EDX = PDE index
000058C7 C1E602              <1>        shl    esi, 2 ; convert PDE index to dword offset
000058CA 01DE                <1>        add    esi, ebx ; add page directory address
                             <1> a_u_pd_0:
000058CC AD                  <1>        lodsd
                             <1>        ;
000058CD 89F3                <1>        mov    ebx, esi ; next PDE address
                             <1>        ;
000058CF A801                <1>        test   al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
000058D1 7513                <1>        jnz    short a_u_pd_2
                             <1>        ;
                             <1>        ; empty PDE (it does not point to valid page table address)
000058D3 E8A2F2FFFF          <1>        call   allocate_page  ; (allocate a new page table)
000058D8 7302                <1>        jnc    short a_u_pd_1 ; OK... now, we have a new page table.
                             <1>        ; cf = 1
                             <1>        ; There is not a free memory page to allocate a new page table !!!
000058DA 5E                  <1>        pop    esi ; *
                             <1> a_u_im_retn:
000058DB C3                  <1>        retn   ; return to 'sysalloc' with 'insufficient memory' error
                             <1>        ;
                             <1> a_u_pd_1: ; clear the new page table content
                             <1>        ; EAX = Physical (base) address of the new page table
000058DC E813F3FFFF          <1>        call   clear_page ; Clear page content
                             <1>        ;
000058E1 0C07                <1>        or     al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
                             <1>                ; set bit 0, bit 1 and bit 2 to 1
                             <1>                ; (present, writable, user)
000058E3 8946FC              <1>        mov    [esi-4], eax
                             <1> a_u_pd_2:
000058E6 662500F0            <1>        and    ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
                             <1>        ; EAX = PHYSICAL (flat) ADDRESS OF THE PAGE TABLE
000058EA 8B3C24              <1>        mov    edi, [esp] ; *
                             <1>        ; EDI = PTE index (of page directory)
                             <1>        ;and   edi, PTE_MASK ; PTE entry index in the page table
                             <1>        ; EBX = next PDE address
000058ED 89FE                <1>        mov    esi, edi ; PTE index in page table (0-1023)
000058EF C1E702              <1>        shl    edi, 2 ; convert PTE index to dword offset
000058F2 01C7                <1>        add    edi, eax ; now, edi points to requested PTE
                             <1> a_u_pt_0:
                             <1>        ; 02/05/2017
000058F4 8B07                <1>        mov    eax, [edi]
                             <1>        ;
000058F6 A801                <1>        test   al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
000058F8 7445                <1>        jz     short a_u_pt_1
                             <1>        ;
000058FA A802                <1>        test   al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
                             <1>                              ; (must be 1)
000058FC 7550                <1>        jnz    short a_u_pt_3
                             <1>        ; Read only -duplicated- page (belongs to a parent or a child)
000058FE 66A90002            <1>          test    ax, PTE_DUPLICATED ; Was this page duplicated
                             <1>                               ; as child's page ?
00005902 7455                <1>        jz     short a_u_pt_4      ; Clear PTE but don't deallocate the page!
                             <1>        ;
                             <1>        ; check the parent's PTE value is read only & same page or not..
                             <1>       ; EDX = page directory entry index (0-1023)
00005904 52                  <1>       push   edx ; **
00005905 53                  <1>       push   ebx ; ***
                             <1>       ; ESI = page table entry index (0-1023)
                             <1>       ;push  esi ; **** ; 20/05/2017
00005906 8B1D[BC030300]      <1>       mov    ebx, [u.ppgdir] ; page directory of the parent process
0000590C 66C1E202            <1>       shl    dx, 2 ; *4
00005910 01D3                <1>       add    ebx, edx ; PTE address,0 (for the parent)
00005912 8B13                <1>       mov    edx, [ebx] ; page table address
00005914 F6C201              <1>       test   dl, PDE_A_PRESENT ; present (valid) or not ?
00005917 7433                <1>       jz     short a_u_pt_2      ; parent process does not use this page
00005919 6681E200F0          <1>       and    dx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
0000591E 66C1E602            <1>       shl    si, 2 ; *4
                             <1>       ; ESI = page table entry offset (0-4092)
00005922 01D6                <1>       add    esi, edx     ; PTE address (for the parent)
00005924 8B1E                <1>       mov    ebx, [esi]
00005926 F6C301              <1>       test   bl, PTE_A_PRESENT ; present or not ?
00005929 7421                <1>       jz     short a_u_pt_2      ; parent process does not use this page
0000592B 662500F0            <1>       and    ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
0000592F 6681E300F0          <1>       and    bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
00005934 39D8                <1>       cmp    eax, ebx     ; parent's and child's pages are same ?
00005936 7514                <1>       jne    short a_u_pt_2      ; not same page
                             <1>                           ; deallocate the child's page
00005938 800E02              <1>         or      byte [esi], PTE_A_WRITE ; convert to writable page (parent)
                             <1>       ;pop   esi ; **** ; 20/05/2017
0000593B 5B                  <1>       pop    ebx ; ***
0000593C 5A                  <1>       pop    edx ; **
0000593D EB1A                <1>       jmp    short a_u_pt_4
                             <1> a_u_pt_1:
0000593F 09C0                <1>       or     eax, eax     ; swapped page ?
00005941 7416                <1>       jz     short a_u_pt_4      ; no
                             <1>                           ; yes
00005943 D1E8                <1>       shr    eax, 1
00005945 E8B5F8FFFF          <1>       call   unlink_swap_block ; Deallocate swapped page block
                             <1>                             ; on the swap disk (or in file)
0000594A EB0D                <1>       jmp    short a_u_pt_4
                             <1> a_u_pt_2:
                             <1>       ;pop   esi ; **** ; 20/05/2017
0000594C 5B                  <1>       pop    ebx ; ***
0000594D 5A                  <1>       pop    edx ; **
                             <1> a_u_pt_3:
0000594E 66A90004            <1>       test   ax, PTE_SHARED      ; shared or direct memory access indicator
00005952 7505                <1>       jnz    short a_u_pt_4      ; AVL bit 1 = 1, do not deallocate this page!
                             <1>       ;
                             <1>       ;and   ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
00005954 E8FFF3FFFF          <1>       call   deallocate_page ; set the mem allocation bit of this page
                             <1>       ;
                             <1> a_u_pt_4:
00005959 89E8                <1>       mov    eax, ebp ; physical address
0000595B 0C07                <1>       or     al, PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER ; 04/03/2017
0000595D AB                  <1>       stosd
0000595E 5E                  <1>       pop    esi ; * ; 20/05/2017
0000595F 49                  <1>       dec    ecx ; remain page count
00005960 7417                <1>       jz     short a_u_pd_5
00005962 81C500100000        <1>       add    ebp, PAGE_SIZE
00005968 46                  <1>       inc    esi ; next PTE (index)
                             <1>       ; 20/05/2017
                             <1>       ;cmp   esi, PAGE_SIZE/4 ; 1024
                             <1>       ;jb    short a_u_pt_0
00005969 6681E6FF03          <1>       and    si, PTE_MASK ; 3FFh (0 to 1023)
0000596E 56                  <1>       push   esi ; *
0000596F 7583                <1>       jnz    short a_u_pt_0 ; > 0 (<1024)
                             <1> a_u_pd_3:
00005971 42                  <1>       inc    edx
                             <1> ;     cmp    edx, 1024
                             <1> ;     jnb    short a_u_pd_4 ; 02/05/2017 (error!, ecx > 0)
00005972 89DE                <1>       mov    esi, ebx ; the next PDE address
00005974 E953FFFFFF          <1>       jmp    a_u_pd_0
                             <1> a_u_pd_4:
                             <1>       ; 02/05/2017
                             <1> ;     stc
                             <1> a_u_pd_5:
                             <1>       ; 20/05/2017
                             <1>       ;pop   edi ; *
00005979 C3                  <1>       retn
                             <1>
                             <1>
                             <1> ; /// End Of MEMORY MANAGEMENT FUNCTIONS ///
                             <1>
                             <1> ;; Data:
                             <1>
                             <1> ; 09/03/2015
                             <1> ;swpq_count: dw 0 ; count of pages on the swap que
                             <1> ;swp_drv:    dd 0 ; logical drive description table address of the swap drive/disk
                             <1> ;swpd_size:  dd 0 ; size of swap drive/disk (volume) in sectors (512 bytes).

                             <1> ;swpd_free:  dd 0 ; free page blocks (4096 bytes) on swap disk/drive (logical)
                             <1> ;swpd_next:  dd 0 ; next free page block
                             <1> ;swpd_last:  dd 0 ; last swap page block
                                 %include 'timer.s'   ; 17/01/2015
                             <1> ; ****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - timer.s
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 15/01/2017
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 17/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Turkish Rational DOS
                             <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
                             <1> ;
                             <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                             <1> ;
                             <1> ; Derived from 'IBM PC-AT' BIOS source code (1985)
                             <1> ; ****************************************************************************
                             <1>
                             <1> ; TRDOS 386  (TRDOS v2.0) Kernel - TIMER & REAL TIME CLOCK (BIOS) FUNCTIONS
                             <1>
                             <1> ; IBM PC-AT BIOS Source Code ('BIOS2.ASM')
                             <1> ; TITLE BIOS2 ---- 06/10/85 BIOS INTERRUPT ROUTINES
                             <1>
                             <1> ;
                             <1> ; ///////// TIMER (& REAL TIME CLOCK) FUNCTIONS //////////////
                             <1>
                             <1> int1Ah:
                             <1>       ; 29/01/2016
                             <1>       ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
0000597A 9C                  <1>       pushfd
0000597B 0E                  <1>       push   cs
0000597C E801000000          <1>       call   TIME_OF_DAY_1
00005981 C3                  <1>       retn
                             <1>
                             <1> ;--- INT  1A H -- (TIME OF DAY) -------------------------------------------------
                             <1> ;       THIS BIOS ROUTINE ALLOWS THE CLOCKS TO BE SET OR READ                 :
                             <1> ;                                                               :
                             <1> ; PARAMETERS:                                                         :
                             <1> ;     (AH) = 00H  READ THE CURRENT SETTING AND RETURN WITH,                   :
                             <1> ;                      (CX) = HIGH PORTION OF COUNT                     :
                             <1> ;                      (DX) = LOW PORTION OF COUNT                      :
                             <1> ;                      (AL) = 0 TIMER HAS NOT PASSED 24 HOURS SINCE LAST READ :
                             <1> ;                             1 IF ON ANOTHER DAY. (RESET TO ZERO AFTER READ) :
                             <1> ;                                                               :
                             <1> ;     (AH) = 01H  SET THE CURRENT CLOCK USING,                                :
                             <1> ;                (CX) = HIGH PORTION OF COUNT                           :
                             <1> ;                (DX) = LOW PORTION OF COUNT.                           :
                             <1> ;                                                               :
                             <1> ;               NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SECOND :
                             <1> ;                            (OR ABOUT 18.2 PER SECOND -- SEE EQUATES)        :
                             <1> ;                                                               :
                             <1> ;     (AH) = 02H  READ THE REAL TIME CLOCK AND RETURN WITH,                   :
                             <1> ;                      (CH) = HOURS IN BCD (00-23)                      :
                             <1> ;                      (CL) = MINUTES IN BCD (00-59)                          :
                             <1> ;                      (DH) = SECONDS IN BCD (00-59)                          :
                             <1> ;                      (DL) = DAYLIGHT SAVINGS ENABLE (00-01)                 :
                             <1> ;                                                               :
                             <1> ;     (AH) = 03H  SET THE REAL TIME CLOCK USING,                        :
                             <1> ;                     (CH) = HOURS IN BCD (00-23)                       :
                             <1> ;                     (CL) = MINUTES IN BCD (00-59)                     :
                             <1> ;                     (DH) = SECONDS IN BCD (00-59)                     :
                             <1> ;                     (DL) = 01 IF DAYLIGHT SAVINGS ENABLE OPTION, ELSE 00.   :
                             <1> ;                                                               :
                             <1> ;             NOTE: (DL) = 00 IF DAYLIGHT SAVINGS TIME ENABLE IS NOT ENABLED. :
                             <1> ;                   (DL) = 01 ENABLES TWO SPECIAL UPDATES THE LAST SUNDAY IN  :
                             <1> ;                APRIL   (1:59:59 --> 3:00:00 AM) AND THE LAST SUNDAY IN      :
                             <1> ;                   OCTOBER (1:59:59 --> 1:00:00 AM) THE FIRST TIME.          :
                             <1> ;                                                               :
                             <1> ;     (AH) = 04H  READ THE DATE FROM THE REAL TIME CLOCK AND RETURN WITH,     :
                             <1> ;                      (CH) = CENTURY IN BCD (19 OR 20)                       :
                             <1> ;                      (CL) = YEAR IN BCD (00-99)                       :
                             <1> ;                      (DH) = MONTH IN BCD (01-12)                      :
                             <1> ;                      (DL) = DAY IN BCD (01-31).                       :
                             <1> ;                                                               :
                             <1> ;     (AH) = 05H  SET THE DATE INTO THE REAL TIME CLOCK USING,                :
                             <1> ;                     (CH) = CENTURY IN BCD (19 OR 20)                        :
                             <1> ;                     (CL) = YEAR IN BCD (00-99)                        :
                             <1> ;                     (DH) = MONTH IN BCD (01-12)                       :
                             <1> ;                     (DL) = DAY IN BCD (01-31).                        :
                             <1> ;                                                               :
                             <1> ;     (AH) = 06H  SET THE ALARM TO INTERRUPT AT SPECIFIED TIME,               :
                             <1> ;                     (CH) = HOURS IN BCD (00-23 (OR FFH))              :
                             <1> ;                     (CL) = MINUTES IN BCD (00-59 (OR FFH))                  :
                             <1> ;                     (DH) = SECONDS IN BCD (00-59 (OR FFH))                  :
                             <1> ;                                                               :
                             <1> ;     (AH) = 07H  RESET THE ALARM INTERRUPT FUNCTION.                         :
                             <1> ;                                                               :
                             <1> ; NOTES: FOR ALL RETURNS CY= 0 FOR SUCCESSFUL OPERATION.                :
                             <1> ;        FOR (AH)= 2, 4, 6 - CARRY FLAG SET IF REAL TIME CLOCK NOT OPERATING. :
                             <1> ;        FOR (AH)= 6 - CARRY FLAG SET IF ALARM ALREADY ENABLED.         :
                             <1> ;        FOR THE ALARM FUNCTION (AH = 6) THE USER MUST SUPPLY A ROUTINE AND   :
                             <1> ;         INTERCEPT THE CORRECT ADDRESS IN THE VECTOR TABLE FOR INTERRUPT 4AH.       :
                             <1> ;         USE 0FFH FOR ANY "DO NOT CARE" POSITION FOR INTERVAL INTERRUPTS.    :
                             <1> ;        INTERRUPTS ARE DISABLED DURING DATA MODIFICATION.                    :
                             <1> ;        AH & AL ARE RETURNED MODIFIED AND NOT DEFINED EXCEPT WHERE INDICATED.      :
                             <1> ;--------------------------------------------------------------------------------
                             <1>
                             <1> ; 15/01/2017
                             <1> ; 14/01/2017
                             <1> ; 07/01/2017
                             <1> ; 02/01/2017
                             <1> ; 29/05/2016
                             <1> ; 29/01/2016
                             <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>
                             <1> ; 29/05/2016
                             <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1> int35h:  ; Date/Time functions
                             <1>
                             <1> TIME_OF_DAY_1:
                             <1>       ;sti                       ; INTERRUPTS BACK ON
                             <1>       ; 29/05/2016
00005982 80642408FE          <1>       and    byte [esp+8], 11111110b   ; clear carry bit of eflags register
                             <1>       ;
00005987 80FC08              <1>       cmp    ah, (RTC_TBE-RTC_TB)/4    ; CHECK IF COMMAND IN VALID RANGE (0-7)
0000598A F5                  <1>       cmc                       ; COMPLEMENT CARRY FOR ERROR EXIT
                             <1>       ; (*) jc short TIME_9            ; EXIT WITH CARRY = 1 IF NOT VALID
```

```
0000598B 721A                <1>       jc     short _TIME_9 ; 29/05/2016
                             <1>
0000598D 1E                  <1>       push   ds
0000598E 56                  <1>       push   esi
0000598F 66BE1000            <1>       mov    si, KDATA          ; kernel data segment
00005993 8EDE                <1>       mov    ds, si
                             <1>
                             <1>       ;;15/01/2017
                             <1>       ; 14/01/2017
                             <1>       ; 02/01/2017
                             <1>       ;;mov  byte [intflg], 35h  ; date & time interrupt
                             <1>       ;sti
                             <1>       ;
00005995 C0E402              <1>       shl    ah, 2              ; convert function to dword offset
00005998 0FB6F4              <1>       movzx  esi, ah                   ; PLACE INTO ADDRESSING REGISTER
                             <1>       ;cli                      ; NO INTERRUPTS DURING TIME FUNCTIONS
0000599B FF96[AD590000]      <1>       call   [esi+RTC_TB]       ; VECTOR TO FUNCTION REQUESTED WITH CY=0
                             <1>                                 ; RETURN WITH CARRY FLAG SET FOR RESULT
                             <1>       ;sti                      ; INTERRUPTS BACK ON
000059A1 B400                <1>       mov    ah, 0              ; CLEAR (AH) TO ZERO
000059A3 5E                  <1>       pop    esi                ; RECOVER USERS REGISTER
000059A4 1F                  <1>       pop    ds                 ; RECOVER USERS SEGMENT SELECTOR
                             <1>
                             <1>       ;;15/01/2017
                             <1>       ; 02/01/2017
                             <1>       ;;mov  byte [ss:intflg], 0 ; 07/01/2017
                             <1>
                             <1> ;TIME_9:
                             <1>                                 ; RETURN WITH CY= 0 IF NO ERROR
                             <1>       ; (*) 29/05/2016
                             <1>       ; (*) retf 4 ; skip eflags on stack
000059A5 7305                <1>       jnc    short _TIME_10
                             <1> _TIME_9:
                             <1>       ; 29/05/2016 -set carry flag on stack-
                             <1>       ; [esp] = EIP
                             <1>       ; [esp+4] = CS
                             <1>       ; [esp+8] = E-FLAGS
000059A7 804C240801          <1>       or     byte [esp+8], 1      ; set carry bit of eflags register
                             <1>       ; [esp+12] = ESP (user)
                             <1>       ; [esp+16] = SS (User)
                             <1> _TIME_10:
000059AC CF                  <1>       iretd
                             <1>
                             <1>       ; (*) 29/05/2016 - 'ref 4' intruction causes to stack fault
                             <1>       ; (OUTER-PRIVILEGE-LEVEL)
                             <1>       ; INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986
                             <1>       ; // RETF instruction:
                             <1>       ;
                             <1>       ; IF OperandMode=32 THEN
                             <1>       ;    Load CS:EIP from stack;
                             <1>       ;    Set CS RPL to CPL;
                             <1>       ;    Increment eSP by 8 plus the immediate offset if it exists;
                             <1>       ;    Load SS:eSP from stack;
                             <1>       ; ELSE (* OperandMode=16 *)
                             <1>       ;    Load CS:IP from stack;
                             <1>       ;    Set CS RPL to CPL;
                             <1>       ;    Increment eSP by 4 plus the immediate offset if it exists;
                             <1>       ;    Load SS:eSP from stack;
                             <1>       ; FI;
                             <1>       ;
                             <1>       ; //
                             <1>                                 ; ROUTINE VECTOR TABLE (AH)=
                             <1> RTC_TB:
000059AD [CD590000]          <1>       dd     RTC_00             ; 0 = READ CURRENT CLOCK COUNT
000059B1 [E0590000]          <1>       dd     RTC_10             ; 1 = SET CLOCK COUNT
000059B5 [EE590000]          <1>       dd     RTC_20             ; 2 = READ THE REAL TIME CLOCK TIME
000059B9 [1D5A0000]          <1>       dd     RTC_30             ; 3 = SET REAL TIME CLOCK TIME
000059BD [5F5A0000]          <1>       dd     RTC_40             ; 4 = READ THE REAL TIME CLOCK DATE
000059C1 [8C5A0000]          <1>       dd     RTC_50             ; 5 = SET REAL TIME CLOCK DATE
000059C5 [D95A0000]          <1>       dd     RTC_60             ; 6 = SET THE REAL TIME CLOCK ALARM
000059C9 [2C5B0000]          <1>       dd     RTC_70             ; 7 = RESET ALARM
                             <1>
                             <1> RTC_TBE      equ    $
                             <1>
                             <1> RTC_00:                         ; READ TIME COUNT
000059CD A0[BC580100]        <1>       mov    al, [TIMER_OFL]           ; GET THE OVERFLOW FLAG
000059D2 C605[BC580100]00    <1>       mov    byte [TIMER_OFL], 0 ; AND THEN RESET THE OVERFLOW FLAG
000059D9 8B0D[B8580100]      <1>         mov     ecx, [TIMER_LH]         ; GET COUNT OF TIME
000059DF C3                  <1>       retn
                             <1>
                             <1> RTC_10:                         ; SET TIME COUNT
000059E0 890D[B8580100]      <1>         mov     [TIMER_LH], ecx         ; SET TIME COUNT
000059E6 C605[BC580100]00    <1>       mov    byte [TIMER_OFL], 0 ; RESET OVERFLOW FLAG
000059ED C3                  <1>       retn                      ; RETURN WITH NO CARRY
                             <1>
                             <1> RTC_20:                         ; GET RTC TIME
000059EE E8EB010000          <1>       call   UPD_IPR                   ; CHECK FOR UPDATE IN PROCESS
000059F3 7227                <1>       jc     short RTC_29       ; EXIT IF ERROR (CY= 1)
                             <1>
000059F5 B000                <1>       mov    al, CMOS_SECONDS   ; SET ADDRESS OF SECONDS
000059F7 E8FD010000          <1>       call   CMOS_READ          ; GET SECONDS
000059FC 88C6                <1>       mov    dh, al             ; SAVE
000059FE B00B                <1>       mov    al, CMOS_REG_B            ; ADDRESS ALARM REGISTER
00005A00 E8F4010000          <1>       call   CMOS_READ          ; READ CURRENT VALUE OF DSE BIT
00005A05 2401                <1>       and    al, 00000001b      ; MASK FOR VALID DSE BIT
00005A07 88C2                <1>       mov    dl, al             ; SET [DL] TO ZERO FOR NO DSE BIT
00005A09 B002                <1>       mov    al, CMOS_MINUTES   ; SET ADDRESS OF MINUTES
00005A0B E8E9010000          <1>       call   CMOS_READ          ; GET MINUTES
00005A10 88C1                <1>       mov    cl, al             ; SAVE
00005A12 B004                <1>         mov     al, CMOS_HOURS          ; SET ADDRESS OF HOURS
00005A14 E8E0010000          <1>       call   CMOS_READ          ; GET HOURS
00005A19 88C5                <1>       mov    ch, al             ; SAVE
00005A1B F8                  <1>       clc                       ; SET CY= 0
```

```
                             <1> RTC_29:
00005A1C C3                  <1>      retn                       ; RETURN WITH RESULT IN CARRY FLAG
                             <1>
                             <1> RTC_30:                         ; SET RTC TIME
00005A1D E8BC010000          <1>      call   UPD_IPR                   ; CHECK FOR UPDATE IN PROCESS
00005A22 7305                <1>      jnc    short RTC_35        ; GO AROUND IF CLOCK OPERATING
00005A24 E817010000          <1>      call   RTC_STA                   ; ELSE TRY INITIALIZING CLOCK
                             <1> RTC_35:
00005A29 88F4                <1>      mov    ah, dh              ; GET TIME BYTE - SECONDS
00005A2B B000                <1>      mov    al, CMOS_SECONDS    ; ADDRESS SECONDS
00005A2D E8E0010000          <1>      call   CMOS_WRITE          ; UPDATE SECONDS
00005A32 88CC                <1>      mov    ah, cl              ; GET TIME BYTE - MINUTES
00005A34 B002                <1>      mov    al, CMOS_MINUTES    ; ADDRESS MINUTES
00005A36 E8D7010000          <1>      call   CMOS_WRITE          ; UPDATE MINUTES
00005A3B 88EC                <1>      mov    ah, ch              ; GET TIME BYTE - HOURS
00005A3D B004                <1>      mov    al, CMOS_HOURS            ; ADDRESS HOURS
00005A3F E8CE010000          <1>      call   CMOS_WRITE          ; UPDATE ADDRESS
                             <1>      ;mov   al, CMOS_REG_B            ; ADDRESS ALARM REGISTER
                             <1>      ;mov   ah, al
00005A44 66B80B0B            <1>      mov    ax, CMOS_REG_B * 257
00005A48 E8AC010000          <1>      call   CMOS_READ           ; READ CURRENT TIME
00005A4D 2462                <1>      and    al, 01100010b       ; MASK FOR VALID BIT POSITIONS
00005A4F 0C02                <1>      or     al, 00000010b       ; TURN ON 24 HOUR MODE
00005A51 80E201              <1>      and    dl, 00000001b       ; USE ONLY THE DSE BIT
00005A54 08D0                <1>      or     al, dl              ; GET DAY LIGHT SAVINGS TIME BIT (OSE)
00005A56 86E0                <1>      xchg   ah, al              ; PLACE IN WORK REGISTER AND GET ADDRESS
00005A58 E8B5010000          <1>      call   CMOS_WRITE          ; SET NEW ALARM SITS
00005A5D F8                  <1>      clc                        ; SET CY= 0
00005A5E C3                  <1>      retn                       ; RETURN WITH CY= 0
                             <1>
                             <1> RTC_40:                         ; GET RTC DATE
00005A5F E87A010000          <1>      call   UPD_IPR                   ; CHECK FOR UPDATE IN PROCESS
00005A64 7225                <1>      jc     short RTC_49        ; EXIT IF ERROR (CY= 1)
                             <1>
00005A66 B007                <1>      mov    al, CMOS_DAY_MONTH  ; ADDRESS DAY OF MONTH
00005A68 E88C010000          <1>      call   CMOS_READ           ; READ DAY OF MONTH
00005A6D 88C2                <1>      mov    dl, al              ; SAVE
00005A6F B008                <1>      mov    al, CMOS_MONTH            ; ADDRESS MONTH
00005A71 E883010000          <1>      call   CMOS_READ           ; READ MONTH
00005A76 88C6                <1>      mov    dh, al              ; SAVE
00005A78 B009                <1>      mov    al, CMOS_YEAR       ; ADDRESS YEAR
00005A7A E87A010000          <1>      call   CMOS_READ           ; READ YEAR
00005A7F 88C1                <1>      mov    cl, al              ; SAVE
00005A81 B032                <1>      mov    al, CMOS_CENTURY    ; ADDRESS CENTURY LOCATION
00005A83 E871010000          <1>      call   CMOS_READ           ; GET CENTURY BYTE
00005A88 88C5                <1>      mov    ch, al              ; SAVE
00005A8A F8                  <1>      clc                        ; SET CY=0
                             <1> RTC_49:
00005A8B C3                  <1>      retn                       ; RETURN WITH RESULTS IN CARRY FLAG
                             <1>
                             <1> RTC_50:                         ; SET RTC DATE
00005A8C E84D010000          <1>      call   UPD_IPR                   ; CHECK FOR UPDATE IN PROCESS
00005A91 7305                <1>      jnc    short RTC_55        ; GO AROUND IF NO ERROR
00005A93 E8A8000000          <1>      call   RTC_STA                   ; ELSE INITIALIZE CLOCK
                             <1> RTC_55:
00005A98 66B80600            <1>      mov    ax, CMOS_DAY_WEEK   ; ADDRESS OF DAY OF WEEK BYTE
00005A9C E871010000          <1>      call   CMOS_WRITE          ; LOAD ZEROS TO DAY OF WEEK
00005AA1 88D4                <1>      mov    ah, dl              ; GET DAY OF MONTH BYTE
00005AA3 B007                <1>      mov    al, CMOS_DAY_MONTH  ; ADDRESS DAY OF MONTH BYTE
00005AA5 E868010000          <1>      call   CMOS_WRITE          ; WRITE OF DAY OF MONTH REGISTER
00005AAA 88F4                <1>      mov    ah, dh              ; GET MONTH
00005AAC B008                <1>      mov    al, CMOS_MONTH            ; ADDRESS MONTH BYTE
00005AAE E85F010000          <1>      call   CMOS_WRITE          ; WRITE MONTH REGISTER
00005AB3 88CC                <1>      mov    ah, cl              ; GET YEAR BYTE
00005AB5 B009                <1>      mov    al, CMOS_YEAR       ; ADDRESS YEAR REGISTER
00005AB7 E856010000          <1>      call   CMOS_WRITE          ; WRITE YEAR REGISTER
00005ABC 88EC                <1>      mov    ah, ch              ; GET CENTURY BYTE
00005ABE B032                <1>      mov    al, CMOS_CENTURY    ; ADDRESS CENTURY BYTE
00005AC0 E84D010000          <1>      call   CMOS_WRITE          ; WRITE CENTURY LOCATION
                             <1>      ;mov   al, CMOS_REG_B            ; ADDRESS ALARM REGISTER
                             <1>      ;mov   ah, al
00005AC5 66B80B0B            <1>      mov    ax, CMOS_REG_B * 257
00005AC9 E82B010000          <1>      call   CMOS_READ           ; READ WIRRENT SETTINGS
00005ACE 247F                <1>      and    al, 07Fh            ; CLEAR 'SET BIT'
00005AD0 86E0                <1>      xchg   ah, al              ; MOVE TO WORK REGISTER
00005AD2 E83B010000          <1>      call   CMOS_WRITE          ; AND START CLOCK UPDATING
00005AD7 F8                  <1>      clc                        ; SET CY= 0
00005AD8 C3                  <1>      retn                       ; RETURN CY=0
                             <1>
                             <1> RTC_60:                         ; SET RTC ALARM
00005AD9 B00B                <1>      mov    al, CMOS_REG_B            ; ADDRESS ALARM
00005ADB E819010000          <1>      call   CMOS_READ           ; READ ALARM REGISTER
00005AE0 A820                <1>      test   al, 20h                   ; CHECK FOR ALARM ALREADY ENABLED
00005AE2 F9                  <1>      stc                        ; SET CARRY IN CASE OF ERROR
00005AE3 7542                <1>      jnz    short RTC_69        ; ERROR EXIT IF ALARM SET
00005AE5 E8F4000000          <1>      call   UPD_IPR                   ; CHECK FOR UPDATE IN PROCESS
00005AEA 7305                <1>      jnc    short RTC_65        ; SKIP INITIALIZATION IF NO ERROR
00005AEC E84F000000          <1>      call   RTC_STA                   ; ELSE INITIALIZE CLOCK
                             <1> RTC_65:
00005AF1 88F4                <1>      mov    ah, dh              ; GET SECONDS BYTE
00005AF3 B001                <1>      mov    al, CMOS_SEC_ALARM  ; ADDRESS THE SECONDS ALARM REGISTER
00005AF5 E818010000          <1>      call   CMOS_WRITE          ; INSERT SECONDS
00005AFA 88CC                <1>      mov    ah, cl              ; GET MINUTES PARAMETER
00005AFC B003                <1>      mov    al, CMOS_MIN_ALARM  ; ADDRESS MINUTES ALARM REGISTER
00005AFE E80F010000          <1>      call   CMOS_WRITE          ; INSERT MINUTES
00005B03 88EC                <1>      mov    ah, ch              ; GET HOURS PARAMETER
00005B05 B005                <1>      mov    al, CMOS_HR_ALARM   ; ADDRESS HOUR ALARM REGISTER
00005B07 E806010000          <1>      call   CMOS_WRITE          ; INSERT HOURS
00005B0C E4A1                <1>      in     al, INTB01          ; READ SECOND INTERRUPT MASK REGISTER
00005B0E 24FE                <1>      and    al, 0FEh            ; ENABLE ALARM TIMER BIT (CY= 0)
00005B10 E6A1                <1>      out    INTB01, al          ; WRITE UPDATED MASK
                             <1>      ;mov   al, CMOS_REG_B            ; ADDRESS ALARM REGISTER
                             <1>      ;mov   ah, al
```

```
00005B12 66B80B0B            <1>       mov    ax, CMOS_REG_B * 257
00005B16 E8DE000000          <1>       call   CMOS_READ           ; READ CURRENT ALARM REGISTER
00005B1B 247F                <1>       and    al, 07Fh            ; ENSURE SET BIT TURNED OFF
00005B1D 0C20                <1>       or     al, 20h                    ; TURN ON ALARM ENABLE
00005B1F 86E0                <1>       xchg   ah, al              ; MOVE MASK TO OUTPUT REGISTER
00005B21 E8EC000000          <1>       call   CMOS_WRITE          ; WRITE NEW ALARM MASK
00005B26 F8                  <1>       clc                        ; SET CY= 0
                             <1> RTC_69:
00005B27 66B80000            <1>       mov    ax, 0               ; CLEAR AX REGISTER
00005B2B C3                  <1>       retn                       ; RETURN WITH RESULTS IN CARRY FLAC
                             <1>
                             <1> RTC_70:                          ; RESET ALARM
                             <1>       ;mov   al, CMOS_REG_B             ; ADDRESS ALARM REGISTER
                             <1>       ;mov   ah, al
00005B2C 66B80B0B            <1>       mov    ax, CMOS_REG_B * 257        ; ADDRESS ALARM REGISTER (TO BOTH AH,AL)
00005B30 E8C4000000          <1>       call   CMOS_READ           ; READ ALARM REGISTER
00005B35 2457                <1>       and    al, 57h                    ; TURN OFF ALARM ENABLE
00005B37 86E0                <1>       xchg   ah, al              ; SAVE DATA AND RECOVER ADDRESS
00005B39 E8D4000000          <1>       call   CMOS_WRITE          ; RESTORE NEW VALUE
00005B3E F8                  <1>       clc                        ; SET CY= 0
00005B3F C3                  <1>       retn                       ; RETURN WITH NO CARRY
                             <1>
                             <1> RTC_STA:                   ; INITIALIZE REAL TIME CLOCK
                             <1>       ;mov   al, CMOS_REG_A             ; ADDRESS REGISTER A AND LOAD DATA MASK
                             <1>       ;mov   ah, 26h
00005B40 66B80A26            <1>       mov    ax, (26h*100h)+CMOS_REG_A
00005B44 E8C9000000          <1>       call   CMOS_WRITE          ; INITIALIZE STATUS REGISTER A
                             <1>       ;mov   al, CMOS_REG_B             ; SET "SET BIT" FOR CLOCK INITIALIZATION
                             <1>       ;mov   ah, 82h
00005B49 66B80B82            <1>       mov    ax, (82h*100h)+CMOS_REG_B
00005B4D E8C0000000          <1>       call   CMOS_WRITE          ; AND 24 HOUR MODE TO REGISTER B
00005B52 B00C                <1>       mov    al, CMOS_REG_C             ; ADDRESS REGISTER C
00005B54 E8A0000000          <1>       call   CMOS_READ           ; READ REGISTER C TO INITIALIZE
00005B59 B00D                <1>       mov    al, CMOS_REG_D             ; ADDRESS REGISTER D
00005B5B E899000000          <1>       call   CMOS_READ           ; READ REGISTER D TO INITIALIZE
00005B60 C3                  <1>       retn
                             <1>
                             <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>
                             <1> ;--- HARDWARE INT  70 H -- ( IRQ LEVEL  8) --------------------------------------
                             <1> ; ALARM INTERRUPT HANDLER (RTC)                                           :
                             <1> ;       THIS ROUTINE HANDLES THE PERIODIC AND ALARM INTERRUPTS FROM THE CMOS  :
                             <1> ;       TIMER. INPUT FREQUENCY IS 1.024 KHZ OR APPROXIMATELY 1024 INTERRUPTS  :
                             <1> ;       EVERY SECOND FOR THE PERIODIC INTERRUPT. FOR THE ALARM FUNCTION,      :
                             <1> ;       THE INTERRUPT WILL OCCUR AT THE DESIGNATED TIME.                :
                             <1> ;                                                                :
                             <1> ;       INTERRUPTS ARE ENABLED WHEN THE EVENT OR ALARM FUNCTION IS ACTIVATED. :
                             <1> ;       FOR THE EVENT INTERRUPT, THE HANDLER WILL DECREMENT THE WAIT COUNTER  :
                             <1> ;       AND WHEN IT EXPIRES WILL SET THE DESIGNATED LOCATION TO 80H. FOR      :
                             <1> ;       THE ALARM INTERRUPT. THE USER MUST PROVIDE A ROUTINE TO INTERCEPT     :
                             <1> ;       THE CORRECT ADDRESS FROM THE VECTOR TABLE INVOKED BY INTERRUPT 4AH    :
                             <1> ;       PRIOR TO SETTING THE REAL TIME CLOCK ALARM (INT 1AH, AH= 06H).        :
                             <1> ;--------------------------------------------------------------------------------
                             <1>
                             <1> RTC_A_INT: ; 07/01/2017
                             <1> ;RTC_INT:                        ; ALARM INTERRUPT
00005B61 1E                  <1>       push   ds                  ; LEAVE INTERRUPTS DISABLED
00005B62 50                  <1>       push   eax                 ; SAVE REGISTERS
00005B63 57                  <1>       push   edi
                             <1> RTC_I_1:                         ; CHECK FOR SECOND INTERRUPT
00005B64 66B88C8B            <1>       mov    ax, 256*(CMOS_REG_B+NMI)+CMOS_REG_C+NMI ; ALARM AND STATUS
00005B68 E670                <1>       out    CMOS_PORT, al       ; WRITE ALARM FLAG MASK ADDRESS
00005B6A 90                  <1>       nop                        ; I/O DELAY
00005B6B EB00                <1>       jmp    short $+2
00005B6D E471                <1>       in     al, CMOS_DATA       ; READ AND RESET INTERRUPT REQUEST FLAGS
00005B6F A860                <1>       test   al, 01100000b       ; CHECK FOR EITHER INTERRUPT PENDING
00005B71 745D                <1>       jz     short  RTC_I_9             ; EXIT IF NOT A VALID RTC INTERRUPT
                             <1>
00005B73 86E0                <1>       xchg   ah, al              ; SAVE FLAGS AND GET ENABLE ADDRESS
00005B75 E670                <1>       out    CMOS_PORT, al       ; WRITE ALARM ENABLE MASK ADDRESS
00005B77 90                  <1>       nop                        ; I/O DELAY
00005B78 EB00                <1>       jmp    short $+2
00005B7A E471                <1>       in     al, CMOS_DATA       ; READ CURRENT ALARM ENABLE MASK
00005B7C 20E0                <1>       and    al, ah              ; ALLOW ONLY SOURCES THAT ARE ENABLED
00005B7E A840                <1>       test   al, 01000000b       ; CHECK FOR PERIODIC INTERRUPT
00005B80 743B                <1>       jz     short RTC_I_5       ; SKIP IF NOT A PERIODIC INTERRUPT
                             <1>
                             <1> ;-----       DECREMENT WAIT COUNT BY INTERRUPT INTERVAL
                             <1>
00005B82 66BF1000            <1>       mov    di, KDATA           ; kernel data segment
00005B86 8EDF                <1>       mov    ds, di
                             <1>
00005B88 812D[B0580100]D003- <1>       sub    dword [RTC_LH], 976 ; DECREMENT COUNT BY 1/1024
00005B90 0000                <1>
00005B92 7329                <1>       jnc    short RTC_I_5       ; SKIP TILL 32 BIT WORD LESS THAN ZERO
                             <1>
                             <1> ;-----       TURN OFF PERIODIC INTERRUPT ENABLE
                             <1>
00005B94 6650                <1>       push   ax                  ; SAVE INTERRUPT FLAG MASK
00005B96 66B88B8B            <1>       mov    ax, 257*(CMOS_REG_B+NMI) ; INTERRUPT ENABLE REGISTER
00005B9A E670                <1>       out    CMOS_PORT, al       ; WRITE ADDRESS TO CMOS CLOCK
00005B9C 90                  <1>       nop                        ; I/O DELAY
00005B9D EB00                <1>       jmp    short $+2
00005B9F E471                <1>       in     al, CMOS_DATA       ; READ CURRENT ENABLES
00005BA1 24BF                <1>       and    al, 0BFh            ; TURN OFF PIE
00005BA3 86C4                <1>       xchg   al, ah              ; GET CMOS ADDRESS AND SAVE VALUE
00005BA5 E670                <1>       out    CMOS_PORT, al       ; ADDRESS REGISTER B
00005BA7 86C4                <1>       xchg   al, ah              ; GET NEW INTERRUPT ENABLE MASK
00005BA9 E671                <1>       out    CMOS_DATA, al       ; SET MASK IN INTERRUPT ENABLE REGISTER
00005BAB C605[B4580100]00    <1>       mov    byte [RTC_WAIT_FLAG], 0   ; SET FUNCTION ACTIVE FLAG OFF
00005BB2 8B3D[B5580100]      <1>       mov    edi, [USER_FLAG]    ; SET UP (DS:DI) TO POINT TO USER FLAG
00005BB8 C60780              <1>       mov    byte [edi], 80h            ; TURN ON USERS FLAG
00005BBB 6658                <1>       pop    ax                  ; GET INTERRUPT SOURCE BACK
```

```
                              <1> RTC_I_5:
00005BBD A820                 <1>       test   al, 00100000b       ; TEST FOR ALARM INTERRUPT
00005BBF 740D                 <1>       jz     short RTC_I_7       ; SKIP USER INTERRUPT CALL IF NOT ALARM
                              <1>
00005BC1 B00D                 <1>       mov    al, CMOS_REG_D             ; POINT TO DEFAULT READ ONLY REGISTER
00005BC3 E670                 <1>       out    CMOS_PORT, al       ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
00005BC5 FB                   <1>       sti                        ; INTERRUPTS BACK ON NOW
00005BC6 52                   <1>       push   edx
00005BC7 E8099E0000           <1>       call   INT4Ah              ; TRANSFER TO USER ROUTINE
00005BCC 5A                   <1>       pop    edx
00005BCD FA                   <1>       cli                        ; BLOCK INTERRUPT FOR RETRY
                              <1> RTC_I_7:                         ; RESTART ROUTINE TO HANDLE DELAYED
00005BCE EB94                 <1>       jmp    short RTC_I_1       ;  ENTRY AND SECOND EVENT BEFORE DONE
                              <1>
                              <1> RTC_I_9:                         ; EXIT - NO PENDING INTERRUPTS
00005BD0 B00D                 <1>       mov    al, CMOS_REG_D             ; POINT TO DEFAULT READ ONLY REGISTER
00005BD2 E670                 <1>       out    CMOS_PORT, al       ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
00005BD4 B020                 <1>       mov    al, EOI                    ; END OF INTERRUPT MASK TO 8259 - 2
00005BD6 E6A0                 <1>       out    INTB00, al          ; TO 8259 - 2
00005BD8 E620                 <1>       out    INTA00,      al            ; TO 8259 - 1
00005BDA 5F                   <1>       pop    edi                 ; RESTORE REGISTERS
00005BDB 58                   <1>       pop    eax
00005BDC 1F                   <1>       pop    ds
00005BDD CF                   <1>       iretd                      ; END OF INTERRUPT
                              <1>
                              <1>
                              <1>       ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
                              <1>       ; 22/08/2014 (Retro UNIX 386 v1)
                              <1>       ; IBM PC/AT BIOS source code ----- 10/06/85 (bios2.asm)
                              <1> UPD_IPR:                         ; WAIT TILL UPDATE NOT IN PROGRESS
00005BDE 51                   <1>       push   ecx
                              <1>
                              <1>       ; 29/05/2016
00005BDF B968110000           <1>       mov    ecx, ((1984+244)*4)/2      ; AWARD BIOS 1999, ATIME.ASM
                              <1>                                  ; 'WAITCPU_CK_UD_STAT'
                              <1>                                  ; (244Us + 1984Us)
                              <1>                                  ; (assume each read takes
                              <1>                                  ;  2 microseconds).
                              <1>       ;mov   ecx, 65535
                              <1>              ;mov cx, 800        ; SET TIMEOUT LOOP COUNT (= 800)
                              <1> UPD_10:
00005BE4 B00A                 <1>       mov    al, CMOS_REG_A             ; ADDRESS STATUS REGISTER A
00005BE6 FA                   <1>       cli                        ; NO TIMER INTERRUPTS DURING UPDATES
00005BE7 E80D000000           <1>       call   CMOS_READ           ; READ UPDATE IN PROCESS FLAG
00005BEC A880                 <1>       test   al, 80h                    ; IF UIP BIT IS ON ( CANNOT READ TIME )
00005BEE 7406                 <1>       jz     short UPD_90        ; EXIT WITH CY= 0 IF CAN READ CLOCK NOW
00005BF0 FB                   <1>       sti                        ; ALLOW INTERRUPTS WHILE WAITING
00005BF1 E2F1                 <1>       loop   UPD_10              ; LOOP TILL READY OR TIMEOUT
00005BF3 31C0                 <1>       xor    eax, eax            ; CLEAR RESULTS IF ERROR
                              <1>              ; xor ax, ax
00005BF5 F9                   <1>       stc                        ; SET CARRY FOR ERROR
                              <1> UPD_90:
00005BF6 59                   <1>       pop    ecx                 ; RESTORE CALLERS REGISTER
00005BF7 FA                   <1>       cli                        ; INTERRUPTS OFF DURING SET
00005BF8 C3                   <1>       retn                       ; RETURN WITH CY FLAG SET
                              <1>
                              <1>
                              <1>       ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
                              <1>       ; 22/08/2014 (Retro UNIX 386 v1)
                              <1>       ; IBM PC/AT BIOS source code ----- 10/06/85 (test4.asm)
                              <1>
                              <1> ;--- CMOS_READ -----------------------------------------------------------------
                              <1> ;            READ BYTE FROM CMOS_SYSTEM CLOCK CONFIGURATION TABLE       :
                              <1> ;                                                                 :
                              <1> ; INPUT: (AL)=     CMOS_TABLE ADDRESS TO BE READ                             :
                              <1> ;            BIT    7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT    :
                              <1> ;            BITS 6-0 = ADDRESS OF TABLE LOCATION TO READ               :
                              <1> ;                                                                 :
                              <1> ; OUTPUT: (AL)     VALUE AT LOCATION (AL) MOVED INTO (AL). IF BIT 7 OF (AL) WAS   :
                              <1> ;            ON THEN NMI LEFT DISABLED, DURING THE CMOS READ BOTH NMI AND   :
                              <1> ;            NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY. :
                              <1> ;            THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND    :
                              <1> ;            THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN.      :
                              <1> ;            ONLY THE (AL) REGISTER AND THE NMI STATE IS CHANGED.       :
                              <1> ;-------------------------------------------------------------------------------
                              <1>
                              <1> CMOS_READ:
00005BF9 9C                   <1>       pushf                      ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
00005BFA D0C0                 <1>       rol    al, 1               ; MOVE NMI BIT TO LOW POSITION
00005BFC F9                   <1>       stc                        ; FORCE NMI BIT ON IN CARRY FLAG
00005BFD D0D8                 <1>       rcr    al, 1               ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
00005BFF FA                   <1>       cli                        ; DISABLE INTERRUPTS
00005C00 E670                 <1>       out    CMOS_PORT, al       ; ADDRESS LOCATION AND DISABLE NMI
                              <1>       ; 29/05/2016
                              <1>       ;nop                       ; I/O DELAY
00005C02 E6EB                 <1>       out    0ebh,al      ; NEWIODELAY ; AWARD BIOS 1999, ATIME.ASM
                              <1>       ;
00005C04 E471                 <1>       in     al, CMOS_DATA       ; READ THE REQUESTED CMOS LOCATION
00005C06 6650                 <1>       push   ax                  ; SAVE (AH) REGISTER VALUE AND CMOS BYTE
                              <1>       ; 15/03/2015 ; IBM PC/XT Model 286 BIOS source code
                              <1>                   ; ----- 10/06/85 (test4.asm)
00005C08 B01E                 <1>       mov    al, CMOS_SHUT_DOWN*2       ; GET ADDRESS OF DEFAULT LOCATION
                              <1>       ;mov   al, CMOS_REG_D*2    ; GET ADDRESS OF DEFAULT LOCATION
00005C0A D0D8                 <1>       rcr    al, 1               ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
00005C0C E670                 <1>       out    CMOS_PORT, al       ; SET DEFAULT TO READ ONLY REGISTER
00005C0E 6658                 <1>       pop    ax                  ; RESTORE (AH) AND (AL), CMOS BYTE
00005C10 9D                   <1>       popf
00005C11 C3                   <1>       retn                       ; RETURN WITH FLAGS RESTORED
                              <1>
                              <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
                              <1>
                              <1> ;--- CMOS_WRITE ----------------------------------------------------------------
                              <1> ;     WRITE BYTE TO CMOS SYSTEM CLOCK CONFIGURATION TABLE               :
                              <1> ;                                                              :
                              <1> ; INPUT: (AL)=     CMOS TABLE ADDRESS TO BE WRITTEN TO                      :
                              <1> ;           BIT    7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT    :
                              <1> ;           BITS 6-0 = ADDRESS OF TABLE LOCATION TO WRITE                :
                              <1> ;      (AH)= NEW VALUE TO BE PLACED IN THE ADDRESSED TABLE LOCATION             :
                              <1> ;                                                              :
                              <1> ; OUTPUT:   VALUE IN (AH) PLACED IN LOCATION (AL) WITH NMI LEFT DISABLED   :
                              <1> ;           IF BIT 7 OF (AL) IS ON, DURING THE CMOS UPDATE BOTH NMI AND    :
                              <1> ;           NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY. :
                              <1> ;           THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND    :
                              <1> ;           THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN.      :
                              <1> ;           ONLY THE CMOS LOCATION AND THE NMI STATE IS CHANGED.       :
                              <1> ;---------------------------------------------------------------------------------
                              <1>
                              <1> CMOS_WRITE:                     ; WRITE (AH) TO LOCATION (AL)
00005C12 9C                      <1>       pushf                     ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
00005C13 6650                    <1>       push   ax                 ; SAVE WORK REGISTER VALUES
00005C15 D0C0                    <1>       rol    al, 1              ; MOVE NMI BIT TO LOW POSITION
00005C17 F9                      <1>       stc                       ; FORCE NMI BIT ON IN CARRY FLAG
00005C18 D0D8                    <1>       rcr    al, 1              ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
00005C1A FA                      <1>       cli                       ; DISABLE INTERRUPTS
00005C1B E670                    <1>       out    CMOS_PORT, al      ; ADDRESS LOCATION AND DISABLE NMI
00005C1D 88E0                    <1>       mov    al, ah             ; GET THE DATA BYTE TO WRITE
00005C1F E671                    <1>       out    CMOS_DATA, al      ; PLACE IN REQUESTED CMOS LOCATION
00005C21 B01E                    <1>       mov    al, CMOS_SHUT_DOWN*2      ; GET ADDRESS OF DEFAULT LOCATION
                              <1>       ;mov   al, CMOS_REG_D*2   ; GET ADDRESS OF DEFAULT LOCATION
00005C23 D0D8                    <1>       rcr    al, 1              ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
00005C25 E670                    <1>       out    CMOS_PORT, al      ; SET DEFAULT TO READ ONLY REGISTER
00005C27 90                      <1>       nop                       ; I/O DELAY
00005C28 E471                    <1>       in     al, CMOS_DATA      ; OPEN STANDBY LATCH
00005C2A 6658                    <1>       pop    ax                 ; RESTORE WORK REGISTERS
00005C2C 9D                      <1>       popf
00005C2D C3                      <1>       retn
                              <1>
                              <1> ; /// End Of TIMER FUNCTIONS ///

00005C2E 90<rept>                    Align 16

                                  gdt:  ; Global Descriptor Table
                                        ; (30/07/2015, conforming cs)
                                        ; (26/03/2015)
                                        ; (24/03/2015, tss)
                                        ; (19/03/2015)
                                        ; (29/12/2013)
                                        ;
00005C30 0000000000000000                  dw 0, 0, 0, 0       ; NULL descriptor
                                        ; 18/08/2014
                                                     ; 8h kernel code segment, base = 00000000h
                                        ;dw 0FFFFh, 0, 9E00h, 00CFh      ; KCODE  ; 30/12/2016
00005C38 FFFF0000009ACF00                  dw 0FFFFh, 0, 9A00h, 00CFh; KCODE
                                                     ; 10h kernel data segment, base = 00000000h
00005C40 FFFF00000092CF00                  dw 0FFFFh, 0, 9200h, 00CFh; KDATA
                                                     ; 1Bh user code segment, base address = 400000h ; CORE
                                        ;dw 0FBFFh, 0, 0FE40h, 00CFh     ; UCODE  ; 30/12/2016
00005C48 FFFB000040FACF00                  dw 0FBFFh, 0, 0FA40h, 00CFh      ; UCODE
                                                     ; 23h user data segment, base address = 400000h ; CORE
00005C50 FFFB000040F2CF00                  dw 0FBFFh, 0, 0F240h, 00CFh      ; UDATA
                                                     ; Task State Segment
00005C58 6700                              dw 0067h ; Limit = 103 ; (104-1, tss size = 104 byte,
                                                            ;  no IO permission in ring 3)
                                  gdt_tss0:
00005C5A 0000                              dw 0  ; TSS base address, bits 0-15
                                  gdt_tss1:
00005C5C 00                                db 0  ; TSS base address, bits 16-23
                                                     ; 49h
00005C5D E9                                db 11101001b ; E9h => P=1/DPL=11/0/1/0/B/1 --> B = Task is busy (1)
00005C5E 00                                db 0 ; G/0/0/AVL/LIMIT=0000 ; (Limit bits 16-19 = 0000) (G=0, 1 byte)
                                  gdt_tss2:
00005C5F 00                                db 0  ; TSS base address, bits 24-31

                                  gdt_end:
                                        ;; 9Eh = 1001 1110b (GDT byte 5) P=1/DPL=00/1/TYPE=1110,
                                                                  ;; Type= 1 (code)/C=1/R=1/A=0
                                              ; P= Present, DPL=0=ring 0,  1= user (0= system)
                                              ; 1= Code C= Conforming, R= Readable, A = Accessed

                                        ;; 9Ah = 1001 1010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,
                                                                  ;; Type= 1 (code)/C=0/R=1/A=0
                                              ; P= Present, DPL=0=ring 0,  1= user (0= system)
                                              ; 1= Code C= non-Conforming, R= Readable, A = Accessed

                                        ;; 92h = 1001 0010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,
                                                                  ;; Type= 0 (data)/E=0/W=1/A=0
                                              ; P= Present, DPL=0=ring 0,  1= user (0= system)
                                              ; 0= Data E= Expansion direction (1= down, 0= up)
                                              ; W= Writeable, A= Accessed

                                        ;; FEh = 1111 1110b (GDT byte 5) P=1/DPL=11/1/TYPE=1110,
                                                                  ;; Type= 1 (code)/C=1/R=1/A=0
                                              ; P= Present, DPL=3=ring 3,  1= user (0= system)
                                              ; 1= Code C= Conforming, R= Readable, A = Accessed

                                        ;; FAh = 1111 1010b (GDT byte 5) P=1/DPL=11/1/TYPE=1010,
                                                                  ;; Type= 1 (code)/C=0/R=1/A=0
                                              ; P= Present, DPL=3=ring 3,  1= user (0= system)
                                              ; 1= Code C= non-Conforming, R= Readable, A = Accessed

                                        ;; F2h = 1111 0010b (GDT byte 5) P=1/DPL=11/1/TYPE=0010,
                                                                  ;; Type= 0 (data)/E=0/W=1/A=0
                                              ; P= Present, DPL=3=ring 3,  1= user (0= system)
                                              ; 0= Data E= Expansion direction (1= down, 0= up)

                                        ;; CFh = 1100 1111b (GDT byte 6) G=1/B=1/0/AVL=0, Limit=1111b (3)
```

```

                                             ;; Limit = FFFFFh (=> FFFFFh+1= 100000h) // bits 0-15, 48-51 //
                                             ;       = 100000h * 1000h (G=1) = 4GB
                                             ;; Limit = FFBFFh (=> FFBFFh+1= FFC00h) // bits 0-15, 48-51 //
                                             ;       = FFC00h * 1000h (G=1) = 4GB - 4MB
                                             ; G= Granularity (1= 4KB), B= Big (32 bit),
                                             ; AVL= Available to programmers

                                 gdtd:
00005C60 2F00                            dw gdt_end - gdt - 1    ; Limit (size)
00005C62 [305C0000]                      dd gdt                  ; Address of the GDT

                                       ; 20/08/2014
                                 idtd:
00005C66 7F02                            dw idt_end - idt - 1    ; Limit (size)
00005C68 [50550100]                      dd idt                  ; Address of the IDT

                                 ; 20/02/2017
                                 ;;; 11/03/2015
                                 %include 'diskdata.s'     ; DISK (BIOS) DATA (initialized)
                             <1> ; ****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - diskdata.s
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 24/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 24/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Turkish Rational DOS
                             <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
                             <1> ;
                             <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                             <1> ; diskdata.inc (11/03/2015)
                             <1> ;
                             <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)
                             <1> ; ****************************************************************************
                             <1>
                             <1> ; Retro UNIX 386 v1 Kernel - DISKDATA.INC
                             <1> ; Last Modification: 11/03/2015
                             <1> ;     (Initialized Disk Parameters Data section for 'DISKIO.INC')
                             <1> ;
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     80286 INTERRUPT LOCATIONS :
                             <1> ;     REFERENCED BY POST & BIOS :
                             <1> ;----------------------------------------
                             <1>
00005C6C [CF5C0000]          <1> DISK_POINTER:      dd     MD_TBL6             ; Pointer to Diskette Parameter Table
                             <1>
                             <1> ; IBM PC-XT Model 286 source code ORGS.ASM (06/10/85) - 14/12/2014
                             <1> ;----------------------------------------------------------------
                             <1> ; DISK_BASE                                          :
                             <1> ;     THIS IS THE SET OF PARAMETERS REQUIRED FOR          :
                             <1> ;     DISKETTE OPERATION. THEY ARE POINTED AT BY THE            :
                             <1> ;     DATA VARIABLE @DISK_POINTER. TO MODIFY THE PARAMETERS,     :
                             <1> ;     BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT        :
                             <1> ;----------------------------------------------------------------
                             <1>
                             <1> ;DISK_BASE:
                             <1> ;     DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
                             <1> ;     DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
                             <1> ;     DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
                             <1> ;     DB     2            ; 512 BYTES/SECTOR
                             <1> ;     ;DB    15           ; EOT (LAST SECTOR ON TRACK)
                             <1> ;     db     18           ; (EOT for 1.44MB diskette)
                             <1> ;     DB     01BH         ; GAP LENGTH
                             <1> ;     DB     0FFH         ; DTL
                             <1> ;     ;DB    054H         ; GAP LENGTH FOR FORMAT
                             <1> ;     db     06ch         ; (for 1.44MB dsikette)
                             <1> ;     DB     0F6H         ; FILL BYTE FOR FORMAT
                             <1> ;     DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
                             <1> ;     DB     8            ; MOTOR START TIME (1/8 SECONDS)
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     ROM BIOS DATA AREAS       :
                             <1> ;----------------------------------------
                             <1>
                             <1> ;DATA       SEGMENT AT 40H            ; ADDRESS= 0040:0000
                             <1>
                             <1> ;@EQUIP_FLAG DW     ?            ; INSTALLED HARDWARE FLAGS
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     DISKETTE DATA AREAS       :
                             <1> ;----------------------------------------
                             <1>
                             <1> ;@SEEK_STATUS      DB     ?            ; DRIVE RECALIBRATION STATUS
                             <1> ;                                ; BIT 3-0 = DRIVE 3-0 RECALIBRATION
                             <1> ;                                ; BEFORE NEXT SEEK IF BIT IS = 0
                             <1> ;@MOTOR_STATUS     DB     ?            ; MOTOR STATUS
                             <1> ;                                ; BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
                             <1> ;                                ; BIT 7 = CURRENT OPERATION IS A WRITE
                             <1> ;@MOTOR_COUNT      DB     ?            ; TIME OUT COUNTER FOR MOTOR(S) TURN OFF
                             <1> ;@DSKETTE_STATUS DB       ?            ; RETURN CODE STATUS BYTE
                             <1> ;                                ; CMD_BLOCK  IN STACK FOR DISK OPERATION
                             <1> ;@NEC_STATUS DB     7 DUP(?)     ; STATUS BYTES FROM DISKETTE OPERATION
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     POST AND BIOS WORK DATA AREA     :
                             <1> ;----------------------------------------
                             <1>
                             <1> ;@INTR_FLAG  DB     ?            ; FLAG INDICATING AN INTERRUPT HAPPENED
                             <1>
                             <1> ;---------------------------------------
                             <1> ;     TIMER DATA AREA           :
                             <1> ;---------------------------------------
                             <1>
                             <1> ; 17/12/2014  (IRQ 0 - INT 08H)
                             <1> ;TIMER_LOW  equ    46Ch         ; Timer ticks (counter)  @ 40h:006Ch
                             <1> ;TIMER_HIGH equ    46Eh         ; (18.2 timer ticks per second)
                             <1> ;TIMER_OFL  equ    470h         ; Timer - 24 hours flag  @ 40h:0070h
                             <1>
                             <1> ;---------------------------------------
                             <1> ;     ADDITIONAL MEDIA DATA            :
                             <1> ;---------------------------------------
                             <1>
                             <1> ;@LASTRATE  DB     ?            ; LAST DISKETTE DATA RATE SELECTED
                             <1> ;@DSK_STATE DB     ?            ; DRIVE 0 MEDIA STATE
                             <1> ;           DB     ?            ; DRIVE 1 MEDIA STATE
                             <1> ;           DB     ?            ; DRIVE 0 OPERATION START STATE
                             <1> ;           DB     ?            ; DRIVE 1 OPERATION START STATE
                             <1> ;@DSK_TRK   DB     ?            ; DRIVE 0 PRESENT CYLINDER
                             <1> ;           DB     ?            ; DRIVE 1 PRESENT CYLINDER
                             <1>
                             <1> ;DATA       ENDS                ; END OF BIOS DATA SEGMENT
                             <1>
                             <1> ;-------------------------------------------------------
                             <1> ;     DRIVE TYPE TABLE                       :
                             <1> ;-------------------------------------------------------
                             <1>             ; 16/02/2015 (unix386.s, 32 bit modifications)
                             <1> DR_TYPE:
00005C70 01                  <1>             DB     01           ;DRIVE TYPE, MEDIA TABLE
                             <1>                 ;DW      MD_TBL1
00005C71 [8E5C0000]          <1>             dd     MD_TBL1
00005C75 82                  <1>             DB     02+BIT7ON
                             <1>             ;DW      MD_TBL2
00005C76 [9B5C0000]          <1>                 dd      MD_TBL2
00005C7A 02                  <1> DR_DEFAULT: DB     02
                             <1>                 ;DW      MD_TBL3
00005C7B [A85C0000]          <1>             dd      MD_TBL3
00005C7F 03                  <1>             DB     03
                             <1>                 ;DW      MD_TBL4
00005C80 [B55C0000]          <1>             dd      MD_TBL4
00005C84 84                  <1>             DB     04+BIT7ON
                             <1>                 ;DW      MD_TBL5
00005C85 [C25C0000]          <1>             dd      MD_TBL5
00005C89 04                  <1>             DB     04
                             <1>                 ;DW      MD_TBL6
00005C8A [CF5C0000]          <1>             dd      MD_TBL6
                             <1> DR_TYPE_E       equ $                   ; END OF TABLE
                             <1> ;DR_CNT            EQU    (DR_TYPE_E-DR_TYPE)/3
                             <1> DR_CNT             equ    (DR_TYPE_E-DR_TYPE)/5
                             <1> ;-------------------------------------------------------
                             <1> ;     MEDIA/DRIVE PARAMETER TABLES                  :
                             <1> ;-------------------------------------------------------
                             <1> ;-------------------------------------------------------
                             <1> ;     360 KB MEDIA IN 360 KB DRIVE                  :
                             <1> ;-------------------------------------------------------
                             <1> MD_TBL1:
00005C8E DF                  <1>       DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00005C8F 02                  <1>       DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00005C90 25                  <1>       DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00005C91 02                  <1>       DB     2            ; 512 BYTES/SECTOR
00005C92 09                  <1>       DB     09           ; EOT (LAST SECTOR ON TRACK)
00005C93 2A                  <1>       DB     02AH         ; GAP LENGTH
00005C94 FF                  <1>       DB     0FFH         ; DTL
00005C95 50                  <1>       DB     050H         ; GAP LENGTH FOR FORMAT
00005C96 F6                  <1>       DB     0F6H         ; FILL BYTE FOR FORMAT
00005C97 0F                  <1>       DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00005C98 08                  <1>       DB     8            ; MOTOR START TIME (1/8 SECONDS)
00005C99 27                  <1>       DB     39           ; MAX. TRACK NUMBER
00005C9A 80                  <1>       DB     RATE_250     ; DATA TRANSFER RATE
                             <1> ;-------------------------------------------------------
                             <1> ;     360 KB MEDIA IN 1.2 MB DRIVE                  :
                             <1> ;-------------------------------------------------------
                             <1> MD_TBL2:
00005C9B DF                  <1>       DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00005C9C 02                  <1>       DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00005C9D 25                  <1>       DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00005C9E 02                  <1>       DB     2            ; 512 BYTES/SECTOR
00005C9F 09                  <1>       DB     09           ; EOT (LAST SECTOR ON TRACK)
00005CA0 2A                  <1>       DB     02AH         ; GAP LENGTH
00005CA1 FF                  <1>       DB     0FFH         ; DTL
00005CA2 50                  <1>       DB     050H         ; GAP LENGTH FOR FORMAT
00005CA3 F6                  <1>       DB     0F6H         ; FILL BYTE FOR FORMAT
00005CA4 0F                  <1>       DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00005CA5 08                  <1>       DB     8            ; MOTOR START TIME (1/8 SECONDS)
00005CA6 27                  <1>       DB     39           ; MAX. TRACK NUMBER
00005CA7 40                  <1>       DB     RATE_300     ; DATA TRANSFER RATE
                             <1> ;-------------------------------------------------------
                             <1> ;     1.2 MB MEDIA IN 1.2 MB DRIVE                  :
                             <1> ;-------------------------------------------------------
                             <1> MD_TBL3:
00005CA8 DF                  <1>       DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00005CA9 02                  <1>       DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00005CAA 25                  <1>       DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00005CAB 02                  <1>       DB     2            ; 512 BYTES/SECTOR
00005CAC 0F                  <1>       DB     15           ; EOT (LAST SECTOR ON TRACK)
00005CAD 1B                  <1>       DB     01BH         ; GAP LENGTH
00005CAE FF                  <1>       DB     0FFH         ; DTL
00005CAF 54                  <1>       DB     054H         ; GAP LENGTH FOR FORMAT
00005CB0 F6                  <1>       DB     0F6H         ; FILL BYTE FOR FORMAT
00005CB1 0F                  <1>       DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00005CB2 08                  <1>       DB     8            ; MOTOR START TIME (1/8 SECONDS)
00005CB3 4F                  <1>       DB     79           ; MAX. TRACK NUMBER
00005CB4 00                  <1>       DB     RATE_500     ; DATA TRANSFER RATE
```

```
                             <1> ;--------------------------------------------------------
                             <1> ;     720 KB MEDIA IN 720 KB DRIVE                  :
                             <1> ;--------------------------------------------------------
                             <1> MD_TBL4:
00005CB5 DF                  <1>       DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00005CB6 02                  <1>       DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00005CB7 25                  <1>       DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00005CB8 02                  <1>       DB     2            ; 512 BYTES/SECTOR
00005CB9 09                  <1>       DB     09           ; EOT (LAST SECTOR ON TRACK)
00005CBA 2A                  <1>       DB     02AH         ; GAP LENGTH
00005CBB FF                  <1>       DB     0FFH         ; DTL
00005CBC 50                  <1>       DB     050H         ; GAP LENGTH FOR FORMAT
00005CBD F6                  <1>       DB     0F6H         ; FILL BYTE FOR FORMAT
00005CBE 0F                  <1>       DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00005CBF 08                  <1>       DB     8            ; MOTOR START TIME (1/8 SECONDS)
00005CC0 4F                  <1>       DB     79           ; MAX. TRACK NUMBER
00005CC1 80                  <1>       DB     RATE_250     ; DATA TRANSFER RATE
                             <1> ;--------------------------------------------------------
                             <1> ;     720 KB MEDIA IN 1.44 MB DRIVE                 :
                             <1> ;--------------------------------------------------------
                             <1> MD_TBL5:
00005CC2 DF                  <1>       DB     11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00005CC3 02                  <1>       DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00005CC4 25                  <1>       DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00005CC5 02                  <1>       DB     2            ; 512 BYTES/SECTOR
00005CC6 09                  <1>       DB     09           ; EOT (LAST SECTOR ON TRACK)
00005CC7 2A                  <1>       DB     02AH         ; GAP LENGTH
00005CC8 FF                  <1>       DB     0FFH         ; DTL
00005CC9 50                  <1>       DB     050H         ; GAP LENGTH FOR FORMAT
00005CCA F6                  <1>       DB     0F6H         ; FILL BYTE FOR FORMAT
00005CCB 0F                  <1>       DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00005CCC 08                  <1>       DB     8            ; MOTOR START TIME (1/8 SECONDS)
00005CCD 4F                  <1>       DB     79           ; MAX. TRACK NUMBER
00005CCE 80                  <1>       DB     RATE_250     ; DATA TRANSFER RATE
                             <1> ;--------------------------------------------------------
                             <1> ;     1.44 MB MEDIA IN 1.44 MB DRIVE                :
                             <1> ;--------------------------------------------------------
                             <1> MD_TBL6:
00005CCF AF                  <1>       DB     10101111B    ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE
00005CD0 02                  <1>       DB     2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00005CD1 25                  <1>       DB     MOTOR_WAIT   ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00005CD2 02                  <1>       DB     2            ; 512 BYTES/SECTOR
00005CD3 12                  <1>       DB     18           ; EOT (LAST SECTOR ON TRACK)
00005CD4 1B                  <1>       DB     01BH         ; GAP LENGTH
00005CD5 FF                  <1>       DB     0FFH         ; DTL
00005CD6 6C                  <1>       DB     06CH         ; GAP LENGTH FOR FORMAT
00005CD7 F6                  <1>       DB     0F6H         ; FILL BYTE FOR FORMAT
00005CD8 0F                  <1>       DB     15           ; HEAD SETTLE TIME (MILLISECONDS)
00005CD9 08                  <1>       DB     8            ; MOTOR START TIME (1/8 SECONDS)
00005CDA 4F                  <1>       DB     79           ; MAX. TRACK NUMBER
00005CDB 00                  <1>       DB     RATE_500     ; DATA TRANSFER RATE
                             <1>
                             <1>
                             <1> ; << diskette.inc >>
                             <1> ; +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                             <1> ;
                             <1> ;----------------------------------------
                             <1> ;     ROM BIOS DATA AREAS        :
                             <1> ;----------------------------------------
                             <1>
                             <1> ;DATA        SEGMENT AT 40H            ; ADDRESS= 0040:0000
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     FIXED DISK DATA AREAS            :
                             <1> ;----------------------------------------
                             <1>
                             <1> ;DISK_STATUS1:     DB     0            ; FIXED DISK STATUS
                             <1> ;HF_NUM:           DB     0            ; COUNT OF FIXED DISK DRIVES
                             <1> ;CONTROL_BYTE:     DB     0            ; HEAD CONTROL BYTE
                             <1> ;@PORT_OFF   DB     ?            ;  RESERVED (PORT OFFSET)
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     ADDITIONAL MEDIA DATA            :
                             <1> ;----------------------------------------
                             <1>
                             <1> ;@LASTRATE   DB     ?            ; LAST DISKETTE DATA RATE SELECTED
                             <1> ;HF_STATUS   DB     0            ; STATUS REGISTER
                             <1> ;HF_ERROR    DB     0            ; ERROR REGISTER
                             <1> ;HF_INT_FLAG DB     0            ; FIXED DISK INTERRUPT FLAG
                             <1> ;HF_CNTRL    DB     0            ; COMBO FIXED DISK/DISKETTE CARD BIT 0=1
                             <1> ;@DSK_STATE  DB     ?            ; DRIVE 0 MEDIA STATE
                             <1> ;            DB     ?            ; DRIVE 1 MEDIA STATE
                             <1> ;            DB     ?            ; DRIVE 0 OPERATION START STATE
                             <1> ;            DB     ?            ; DRIVE 1 OPERATION START STATE
                             <1> ;@DSK_TRK    DB     ?            ; DRIVE 0 PRESENT CYLINDER
                             <1> ;            DB     ?            ; DRIVE 1 PRESENT CYLINDER
                             <1>
                             <1> ;DATA        ENDS                ; END OF BIOS DATA SEGMENT
                             <1> ;
                             <1> ; +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                             <1>
                             <1> ERR_TBL:
00005CDC E0                  <1>       db     NO_ERR
00005CDD 024001BB            <1>       db     BAD_ADDR_MARK,BAD_SEEK,BAD_CMD,UNDEF_ERR
00005CE1 04BB100A            <1>       db     RECORD_NOT_FND,UNDEF_ERR,BAD_ECC,BAD_SECTOR
                             <1>
                             <1> ; 17/12/2014 (mov ax, [cfd])
                             <1> ; 11/12/2014
00005CE5 00                  <1> cfd:        db 0                ; current floppy drive (for GET_PARM)
                             <1> ; 17/12/2014                    ; instead of 'DISK_POINTER'
00005CE6 01                  <1> pfd:        db 1                ; previous floppy drive (for GET_PARM)
                             <1>                                 ; (initial value of 'pfd
                             <1>                                 ; must be different then 'cfd' value
                             <1>                                 ; to force updating/initializing
                             <1>                                 ; current drive parameters)
00005CE7 90                  <1> align 2
                             <1>
00005CE8 F001                <1> HF_PORT:    dw     1F0h  ; Default = 1F0h
                             <1>                          ; (170h)
00005CEA F603                <1> HF_REG_PORT: dw    3F6h  ; HF_PORT + 206h
                             <1>
                             <1> ; 05/01/2015
00005CEC 00                  <1> hf_m_s:         db      0     ; (0 = Master, 1 = Slave)
                             <1>
                             <1> ; *****************************************************************************

00005CED 90                      Align 2

                                 ; 04/11/2014 (Retro UNIX 386 v1)
00005CEE 0000                    mem_1m_1k:   dw 0  ; Number of contiguous KB between
                                                      ; 1 and 16 MB, max. 3C00h = 15 MB.
00005CF0 0000                    mem_16m_64k: dw 0  ; Number of contiguous 64 KB blocks
                                                ; between 16 MB and 4 GB.

                                 ; 12/11/2014 (Retro UNIX 386 v1)
00005CF2 00                      boot_drv:    db 0 ; boot drive number (physical)
                                 ; 24/11/2014
00005CF3 00                      drv:       db 0
00005CF4 00                      last_drv:    db 0 ; last hdd
00005CF5 00                      hdc:         db 0  ; number of hard disk drives
                                                  ; (present/detected)

                                 ; 24/11/2014 (Retro UNIX 386 v1)
                                 ; Physical drive type & flags
00005CF6 00                      fd0_type:    db 0  ; floppy drive type
00005CF7 00                      fd1_type:    db 0  ; 4 = 1.44 Mb, 80 track, 3.5" (18 spt)
                                                  ; 6 = 2.88 Mb, 80 track, 3.5" (36 spt)
                                                  ; 3 = 720 Kb, 80 track, 3.5" (9 spt)
                                                  ; 2 = 1.2 Mb, 80 track, 5.25" (15 spt)
                                                  ; 1 = 360 Kb, 40 track, 5.25" (9 spt)
00005CF8 00                      hd0_type:    db 0  ; EDD status for hd0 (bit 7 = present flag)
00005CF9 00                      hd1_type:    db 0  ; EDD status for hd1 (bit 7 = present flag)
00005CFA 00                      hd2_type:    db 0  ; EDD status for hd2 (bit 7 = present flag)
00005CFB 00                      hd3_type:    db 0  ; EDD status for hd3 (bit 7 = present flag)
                                                  ; bit 0 - Fixed disk access subset supported
                                                  ; bit 1 - Drive locking and ejecting
                                                  ; bit 2 - Enhanced disk drive support
                                                  ; bit 3 = Reserved (64 bit EDD support)
                                                  ; (If bit 0 is '1' Retro UNIX 386 v1
                                                  ; will interpret it as 'LBA ready'!)

                                 ; 11/03/2015 - 10/07/2015
00005CFC 000000000000000000-     drv.cylinders: dw 0,0,0,0,0,0,0
00005D05 0000000000
00005D0A 000000000000000000-     drv.heads:     dw 0,0,0,0,0,0,0
00005D13 0000000000
00005D18 000000000000000000-     drv.spt:       dw 0,0,0,0,0,0,0
00005D21 0000000000
00005D26 000000000000000000-     drv.size:      dd 0,0,0,0,0,0,0
00005D2F 000000000000000000-
00005D38 000000000000000000-
00005D41 00
00005D42 00000000000000          drv.status:    db 0,0,0,0,0,0,0
00005D49 00000000000000          drv.error:     db 0,0,0,0,0,0,0

                                 Align 2

                                 ;;; 11/03/2015
                                 %include 'kybdata.s'      ; KEYBOARD (BIOS) DATA
                             <1> ; ****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - kybdata.s
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 17/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 17/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Turkish Rational DOS
                             <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
                             <1> ;
                             <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                             <1> ; kybdata.inc (11/03/2015)
                             <1> ;
                             <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)
                             <1> ; ****************************************************************************
                             <1>
                             <1> ; Retro UNIX 386 v1 Kernel - KYBDATA.INC
                             <1> ; Last Modification: 11/03/2015
                             <1> ;            (Data Section for 'KEYBOARD.INC')
                             <1> ;
                             <1> ; ///////// KEYBOARD DATA //////////////
                             <1>
                             <1> ; 05/12/2014
                             <1> ; 04/12/2014 (derived from pc-xt-286 bios source code -1986-)
                             <1> ; 03/06/86  KEYBOARD BIOS
                             <1>
                             <1> ;--------------------------------------------------------------------------------
                             <1> ;     KEY IDENTIFICATION SCAN TABLES
                             <1> ;--------------------------------------------------------------------------------
                             <1>
                             <1> ;-----      TABLES FOR ALT CASE ------------
                             <1> ;-----      ALT-INPUT-TABLE
00005D50 524F50514B          <1> K30:  db     82,79,80,81,75
00005D55 4C4D474849          <1>       db     76,77,71,72,73            ; 10 NUMBER ON KEYPAD
                             <1> ;-----      SUPER-SHIFT-TABLE
```

```
00005D5A 101112131415         <1>       db     16,17,18,19,20,21   ; A-Z TYPEWRITER CHARS
00005D60 161718191E1F         <1>       db     22,23,24,25,30,31
00005D66 202122232425         <1>       db     32,33,34,35,36,37
00005D6C 262C2D2E2F30         <1>       db     38,44,45,46,47,48
00005D72 3132                 <1>       db     49,50
                              <1>
                              <1> ;-----       TABLE OF SHIFT KEYS AND MASK VALUES
                              <1> ;-----       KEY_TABLE
00005D74 52                   <1> _K6:    db      INS_KEY                  ; INSERT KEY
00005D75 3A4546381D           <1>       db     CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
00005D7A 2A36                 <1>         db      LEFT_KEY,RIGHT_KEY
                              <1> _K6L    equ     $-_K6
                              <1>
                              <1> ;-----       MASK_TABLE
00005D7C 80                   <1> _K7:    db      INS_SHIFT                ; INSERT MODE SHIFT
00005D7D 4020100804           <1>       db     CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
00005D82 0201                 <1>       db     LEFT_SHIFT,RIGHT_SHIFT
                              <1>
                              <1> ;-----       TABLES FOR CTRL CASE             ;---- CHARACTERS ------
00005D84 1BFF00FFFFFF         <1> _K8:  db     27,-1,0,-1,-1,-1    ; Esc, 1, 2, 3, 4, 5
00005D8A 1EFFFFFFFF1F         <1>       db     30,-1,-1,-1,-1,31   ; 6, 7, 8, 9, 0, -
00005D90 FF7FFF111705         <1>       db     -1,127,-1,17,23,5   ; =, Bksp, Tab, Q, W, E
00005D96 12141915090F         <1>       db     18,20,25,21,9,15    ; R, T, Y, U, I, O
00005D9C 101B1D0AFF01         <1>       db     16,27,29,10,-1,1    ; P, [, ], Enter, Ctrl, A
00005DA2 13040607080A         <1>       db     19,4,6,7,8,10       ; S, D, F, G, H, J
00005DA8 0B0CFFFFFFFF         <1>       db     11,12,-1,-1,-1,-1   ; K, L, :, ', `, LShift
00005DAE 1C1A18031602         <1>       db     28,26,24,3,22,2           ; Bkslash, Z, X, C, V, B
00005DB4 0E0DFFFFFFFF         <1>       db     14,13,-1,-1,-1,-1   ; N, M, ,, ., /, RShift
00005DBA 96FF20FF             <1>       db     150,-1,' ',-1       ; *, ALT, Spc, CL
                              <1>       ;                          ;----- FUNCTIONS ------
00005DBE 5E5F60616263         <1>       db     94,95,96,97,98,99   ; F1 - F6
00005DC4 64656667FFFF         <1>       db     100,101,102,103,-1,-1     ; F7 - F10, NL, SL
00005DCA 778D848E738F         <1>       db     119,141,132,142,115,143    ; Home, Up, PgUp, -, Left, Pad5
00005DD0 749075917692         <1>       db     116,144,117,145,118,146 ; Right, +, End, Down, PgDn, Ins
00005DD6 93FFFFFF898A         <1>       db     147,-1,-1,-1,137,138      ; Del, SysReq, Undef, WT, F11, F12
                              <1>
                              <1> ;-----       TABLES FOR LOWER CASE ----------
00005DDC 1B3132333435363738-  <1> K10:  db     27,'1234567890-=',8,9
00005DE5 39302D3D0809         <1>
00005DEB 71776572747975696F-  <1>       db     'qwertyuiop[]',13,-1,'asdfghjkl;',39
00005DF4 705B5D0DFF61736466-  <1>
00005DFD 67686A6B6C3B27       <1>
00005E04 60FF5C7A786376626E-  <1>       db     96,-1,92,'zxcvbnm,./',-1,'*',-1,' ',-1
00005E0D 6D2C2E2FFF2AFF20FF   <1>
                              <1> ;-----       LC TABLE SCAN
00005E16 3B3C3D3E3F           <1>       db     59,60,61,62,63            ; BASE STATE OF F1 - F10
00005E1B 4041424344           <1>       db     64,65,66,67,68
00005E20 FFFF                 <1>       db     -1,-1               ; NL, SL
                              <1>
                              <1> ;-----       KEYPAD TABLE
00005E22 474849FF4BFF         <1> K15:  db     71,72,73,-1,75,-1   ; BASE STATE OF KEYPAD KEYS
00005E28 4DFF4F50515253       <1>       db     77,-1,79,80,81,82,83
00005E2F FFFF5C8586           <1>       db     -1,-1,92,133,134    ; SysRq, Undef, WT, F11, F12
                              <1>
                              <1> ;-----       TABLES FOR UPPER CASE ----------
00005E34 1B21402324255E262A-  <1> K11:  db     27,'!@#$%',94,'&*()_+',8,0
00005E3D 28295F2B0800         <1>
00005E43 51574552545955494F-  <1>       db     'QWERTYUIOP{}',13,-1,'ASDFGHJKL:"'
00005E4C 507B7D0DFF41534446-  <1>
00005E55 47484A4B4C3A22       <1>
00005E5C 7EFF7C5A584356424E-  <1>       db     126,-1,'|ZXCVBNM<>?',-1,'*',-1,' ',-1
00005E65 4D3C3E3FFF2AFF20FF   <1>
                              <1> ;-----       UC TABLE SCAN
00005E6E 5455565758           <1> K12:  db     84,85,86,87,88            ; SHIFTED STATE OF F1 - F10
00005E73 595A5B5C5D           <1>       db     89,90,91,92,93
00005E78 FFFF                 <1>       db     -1,-1               ; NL, SL
                              <1>
                              <1> ;-----       NUM STATE TABLE
00005E7A 3738392D3435362B31-  <1> K14:  db     '789-456+1230.'           ; NUMLOCK STATE OF KEYPAD KEYS
00005E83 3233302E             <1>
                              <1>       ;
00005E87 FFFF7C8788           <1>       db     -1,-1,124,135,136   ; SysRq, Undef, WT, F11, F12
                              <1>
                              <1> ; 26/08/2014
                              <1> ; Retro UNIX 8086 v1 - UNIX.ASM (03/03/2014)
                              <1> ; Derived from IBM "pc-at"
                              <1> ; rombios source code (06/10/1985)
                              <1> ; 'dseg.inc'
                              <1>
                              <1> ;---------------------------------------;
                              <1> ;     SYSTEM DATA AREA          ;
                              <1> ;----------------------------------------
00005E8C 00                   <1> BIOS_BREAK   db     0            ; BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
                              <1>
                              <1> ;----------------------------------------
                              <1> ;     KEYBOARD DATA AREAS       ;
                              <1> ;----------------------------------------
                              <1>
00005E8D 00                   <1> KB_FLAG            db     0            ; KEYBOARD SHIFT STATE AND STATUS FLAGS
00005E8E 00                   <1> KB_FLAG_1    db     0            ; SECOND BYTE OF KEYBOARD STATUS
00005E8F 00                   <1> KB_FLAG_2    db     0            ; KEYBOARD LED FLAGS
00005E90 00                   <1> KB_FLAG_3    db     0            ; KEYBOARD MODE STATE AND TYPE FLAGS
00005E91 00                   <1> ALT_INPUT    db     0            ; STORAGE FOR ALTERNATE KEY PAD ENTRY
00005E92 [A25E0000]           <1> BUFFER_START dd     KB_BUFFER    ; OFFSET OF KEYBOARD BUFFER START
00005E96 [C25E0000]           <1> BUFFER_END   dd     KB_BUFFER + 32      ; OFFSET OF END OF BUFFER
00005E9A [A25E0000]           <1> BUFFER_HEAD  dd     KB_BUFFER    ; POINTER TO HEAD OF KEYBOARD BUFFER
00005E9E [A25E0000]           <1> BUFFER_TAIL  dd     KB_BUFFER    ; POINTER TO TAIL OF KEYBOARD BUFFER
                              <1> ; ------     HEAD = TAIL  INDICATES THAT THE BUFFER IS EMPTY
00005EA2 0000<rept>           <1> KB_BUFFER    times  16 dw 0            ; ROOM FOR 16 SCAN CODE ENTRIES
                              <1>
                              <1> ; /// End Of KEYBOARD DATA ///
                                  %include 'vidata.s'       ; VIDEO (BIOS) DATA
                              <1> ; ******************************************************************************
                              <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - vidata.s
                              <1> ; ----------------------------------------------------------------------------
                              <1> ; Last Update: 31/07/2016
                              <1> ; ----------------------------------------------------------------------------
                              <1> ; Beginning: 16/01/2016
                              <1> ; ----------------------------------------------------------------------------
                              <1> ; Assembler: NASM version 2.11 (trdos386.s)
                              <1> ; ----------------------------------------------------------------------------
                              <1> ; Turkish Rational DOS
                              <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
                              <1> ;
                              <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                              <1> ; vidata.inc (11/03/2015)
                              <1> ;
                              <1> ; Derived from 'IBM PC-AT' BIOS source code (1985)
                              <1> ; ****************************************************************************
                              <1>
                              <1> ; Retro UNIX 386 v1 Kernel - VIDATA.S
                              <1> ; Last Modification: 11/03/2015
                              <1> ;               (Data section for 'VIDEO.INC')
                              <1> ;
                              <1> ; ///////// VIDEO DATA ///////////////
                              <1>
                              <1> ;----------------------------------------
                              <1> ;     VIDEO DISPLAY DATA AREA          ;
                              <1> ;----------------------------------------
00005EC2 03                   <1> CRT_MODE:    db     3      ; CURRENT DISPLAY MODE (TYPE)
00005EC3 29                   <1> CRT_MODE_SET:      db     29h    ; CURRENT SETTING OF THE 3X8 REGISTER
                              <1>                           ; (29h default setting for video mode 3)
                              <1>                           ; Mode Select register Bits
                              <1>                           ;   BIT 0 - 80x25 (1), 40x25 (0)
                              <1>                           ;   BIT 1 - ALPHA (0), 320x200 GRAPHICS (1)
                              <1>                           ;   BIT 2 - COLOR (0), BW (1)
                              <1>                           ;   BIT 3 - Video Sig. ENABLE (1), DISABLE (0)
                              <1>                           ;   BIT 4 - 640x200 B&W Graphics Mode (1)
                              <1>                           ;   BIT 5 - ALPHA mode BLINKING (1)
                              <1>                           ;   BIT 6, 7 - Not Used
                              <1>
                              <1> ; Mode 0 - 2Ch = 101100b  ; 40x25 text, 16 gray colors
                              <1> ; Mode 1 - 28h = 101000b  ; 40x25 text, 16 fore colors, 8 back colors
                              <1> ; Mode 2 - 2Dh = 101101b  ; 80x25 text, 16 gray colors
                              <1> ; Mode 3 - 29h = 101001b  ; 80x25 text, 16 fore color, 8 back color
                              <1> ; Mode 4 - 2Ah = 101010b  ; 320x200 graphics, 4 colors
                              <1> ; Mode 5 - 2Eh = 101110b  ; 320x200 graphics, 4 gray colors
                              <1> ; Mode 6 - 1Eh = 011110b  ; 640x200 graphics, 2 colors
                              <1> ; Mode 7 - 29h = 101001b  ; 80x25 text, black & white colors
                              <1> ; Mode & 37h = Video signal OFF
                              <1>
                              <1> ; 24/06/2016
00005EC4 50                   <1> CRT_COLS:    db     80     ; Number of columns
                              <1>
                              <1> ; 01/07/2016
00005EC5 00                   <1> CRT_PALETTE: db     0      ; Current palette setting
                              <1>
                              <1> ; 03/07/2016
00005EC6 10                   <1> CHAR_HEIGHT: db     16     ; Default character height
00005EC7 60                   <1> VGA_VIDEO_CTL:     db     60h    ; ROM BIOS DATA AREA Offset 87h
00005EC8 F9                   <1> VGA_SWITCHES:      db     0F9h   ; Feature Bit Switches (the basic screen)
00005EC9 51                   <1> VGA_MODESET_CTL: db       051h   ; Basic mode set options (VGA video flags)
                              <1>                           ; ROM BIOS DATA AREA Offset 89h
                              <1>                           ; Bit 7, 4 : Mode
                              <1>                           ;        01 : 400-line mode
                              <1>                           ; Bit 6      : Display switch enabled =  1
                              <1>                           ; Bit 5      : Reserved  = 0
                              <1>                           ; Bit 3      : Default palette loading
                              <1>                           ;        disabled = 0
                              <1>                           ; Bit 2 : Color monitor = 0
                              <1>                           ; Bit 1 = Gray scale summing
                              <1>                           ;        disabled = 0
                              <1>                           ; Bit 0 = VGA active = 1
00005ECA 19                   <1> VGA_ROWS:    db     25
                              <1>
                              <1> ; 16/01/2016
                              <1> chr_attrib:  ; Character color/attributes for viode pages (0 to 7)
00005ECB 0707070707070707     <1>       db     07h, 07h, 07h, 07h, 07h, 07h, 07h, 07h
                              <1> ; 30/01/2016
                              <1> vmode:
00005ED3 0303030303030303     <1>       db     3,3,3,3,3,3,3,3 ; video modes for pseudo screens
                              <1>
                              <1> CURSOR_MODE: ; cursor start (ch) = 14, cursor end (cl) = 15
00005EDB 0F0E                 <1>       db     15, 14 ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
                              <1>
                              <1> ;align 4
                              <1> ;VGA_BASE: ; 26/07/2016
                              <1> ;     dd      0B8000h  ; (Mode < 0Dh) or 0A0000h (mode >= 0Dh)
                              <1>
00005EDD 90                   <1> align 2
                              <1>
                              <1> vga_modes:
                              <1>       ; 25/07/2016
                              <1>       ; 09/07/2016
                              <1>       ; 03/07/2016
                              <1>       ; valid (implemented) video modes (>7, extension to IBM PC CGA modes)
00005EDE 0302010007040506     <1>       db     03h, 02h, 01h, 00h, 07h, 04h, 05h, 06h
                              <1> vga_g_modes: ; 31/07/2016
00005EE6 13F0126A0D0E1011     <1>       db     13h, 0F0h, 12h, 6Ah, 0Dh, 0Eh, 10h, 11h
                              <1> vga_mode_count equ $ - vga_modes
                              <1> vga_g_mode_count equ $ - vga_g_modes
                              <1>
                              <1> vga_mode_tbl_ptr:
                              <1>       ; 25/07/2016
00005EEE [4E5F0000]           <1>       dd     vga_mode_03h
00005EF2 [4E5F0000]           <1>       dd     vga_mode_03h ; mode 02h -> mode 03h
```

```
00005EF6 [8E5F0000]          <1>       dd     vga_mode_01h
00005EFA [8E5F0000]          <1>       dd     vga_mode_01h ; mode 00h -> mode 01h
                             <1>       ;dd    vga_mode_07h
00005EFE [4E5F0000]          <1>       dd     vga_mode_03h ; mode 07h -> mode 03h
00005F02 [CE5F0000]          <1>       dd     vga_mode_04h
00005F06 [CE5F0000]          <1>       dd     vga_mode_04h ; mode 05h -> mode 04h
00005F0A [0E600000]          <1>       dd     vga_mode_06h
00005F0E [4E600000]          <1>       dd     vga_mode_13h
00005F12 [8E600000]          <1>       dd     vga_mode_F0h
00005F16 [CE600000]          <1>       dd     vga_mode_12h
00005F1A [0E610000]          <1>       dd     vga_mode_6Ah
00005F1E [4E610000]          <1>       dd     vga_mode_0Dh
00005F22 [8E610000]          <1>       dd     vga_mode_0Eh
00005F26 [CE610000]          <1>       dd     vga_mode_10h
00005F2A [0E620000]          <1>       dd     vga_mode_11h
                             <1>
                             <1> vga_memmodel:
                             <1>       ; 25/07/2016
                             <1>       ; 07/07/2016
                             <1>       CTEXT  equ 0
                             <1>       ;MTEXT equ 1
                             <1>       MTEXT  equ 0 ; mode 07h -> mode 03h
                             <1>       CGA    equ 2
                             <1>       LINEAR8 equ 5
                             <1>       PLANAR4      equ 4
                             <1>       PLANAR1      equ 3
00005F2E 0000000000020202    <1>       db     CTEXT, CTEXT, CTEXT, CTEXT, MTEXT, CGA, CGA, CGA
                             <1> vga_g_memmodel: ; 31/07/2016
00005F36 0504040404040403    <1>       db     LINEAR8, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR1
                             <1> ;vga_pixbits:
                             <1> ;     ; 25/07/2016
                             <1> ;     ; 08/07/2016
                             <1> ;     db     4, 4, 4, 4, 4, 2, 2, 1, 8, 4, 4, 4, 4, 4, 4, 1
                             <1> vga_dac_s:
00005F3E 020202020001010103- <1>       db     2, 2, 2, 2, 0, 1, 1, 1, 3, 3, 2, 2, 1, 1, 2, 2
00005F47 03020201010202      <1>
                             <1>
                             <1> vga_params:
                             <1>       ; 25/07/2016
                             <1>       ; 19/07/2016
                             <1>       ; 03/07/2016
                             <1>       ; derived from 'Plex86/Bochs VGABios' source code
                             <1>       ; vgabios-0.7a (2011)
                             <1>       ; by the LGPL VGABios Developers Team (2001-2008)
                             <1>       ; 'vgatables.h'
                             <1>       ; Oracle VirtualBox 5.0.24 VGABios Source Code
                             <1>       ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
                             <1>       ;
                             <1> vga_mode_03h:  ; mode 03h, 80*25 text, CGA colors
00005F4E 5018100010          <1>       db     80, 24, 16, 00h, 10h ; tw, th-1, ch, slength (5)
00005F53 00030002            <1>       db     00h, 03h, 00h, 02h ; sequ regs (4)
00005F57 67                  <1>       db     67h    ; misc reg (1)
00005F58 5F4F50825581BF1F    <1>       db     5Fh, 4Fh, 50h, 82h, 55h, 81h, 0BFh, 1Fh
00005F60 004F                <1>       db     00h, 4Fh
                             <1> vga_p_cm_pos equ $ - vga_mode_03h
00005F62 0D0E00000000        <1>       db     0Dh, 0Eh, 00h, 00h, 00h, 00h
00005F68 9C8E8F281F96B9A3    <1>       db     9Ch, 8Eh, 8Fh, 28h, 1Fh, 96h, 0B9h, 0A3h
00005F70 FF                  <1>       db     0FFh   ; crtc_regs (25)
00005F71 0001020304051407    <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
00005F79 38393A3B3C3D3E3F    <1>       db     38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
00005F81 0C000F08            <1>       db     0Ch, 00h, 0Fh, 08h  ; actl regs (20)
00005F85 0000000000100E0FFF  <1>       db     00h, 00h, 00h, 00h, 00h, 10h, 0Eh, 0Fh, 0FFh ; grdc regs (9)
                             <1> vga_mode_01h:      ; mode 01h, 40*25 text, CGA colors
00005F8E 2818100008          <1>       db     40, 24, 16, 00h, 08h ; tw, th-1, ch, slength
00005F93 08030002            <1>       db     08h, 03h, 00h, 02h  ; sequ regs
00005F97 67                  <1>       db     67h    ; misc reg
00005F98 2D2728902BA0BF1F    <1>       db     2Dh, 27h, 28h, 90h, 2Bh, 0A0h, 0BFh, 1Fh
00005FA0 004F0D0E00000000    <1>       db     00h, 4Fh, 0Dh, 0Eh, 00h, 00h, 00h, 00h
00005FA8 9C8E8F141F96B9A3    <1>       db     9Ch, 8Eh, 8Fh, 14h, 1Fh, 96h, 0B9h, 0A3h
00005FB0 FF                  <1>       db     0FFh   ; crtc_regs
00005FB1 0001020304051407    <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
00005FB9 38393A3B3C3D3E3F    <1>       db     38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
00005FC1 0C000F08            <1>       db     0Ch, 00h, 0Fh, 08h  ; actl regs
00005FC5 0000000000100E0FFF  <1>       db     00h, 00h, 00h, 00h, 00h, 10h, 0Eh, 0Fh, 0FFh ; grdc regs
                             <1> ;vga_mode_07h:      ; mode 07h, 80*25 text, mono color
                             <1> ;     db     80, 24, 16, 00h, 10h  ; tw, th-1, ch, slength
                             <1> ;     db     00h, 03h, 00h, 02h ; sequ regs
                             <1> ;     db     66h    ; misc reg
                             <1> ;     db     5Fh, 4Fh, 50h, 82h, 55h, 81h, 0BFh, 1Fh
                             <1> ;     db     00h, 4Fh, 0Dh, 0Eh, 00h, 00h, 00h, 00h
                             <1> ;     db     9Ch, 8Eh, 8Fh, 28h, 0Fh, 96h, 0B9h, 0A3h
                             <1> ;     db     0FFh   ; crtc regs
                             <1> ;     db     00h, 08h, 08h, 08h, 08h, 08h, 08h, 08h
                             <1> ;     db     10h, 18h, 18h, 18h, 18h, 18h, 18h, 18h
                             <1> ;     db     0Eh, 00h, 0Fh, 08h  ; actl regs
                             <1> ;     db     00h, 00h, 00h, 00h, 00h, 10h, 0Ah, 0Fh, 0FFh ; grdc regs
                             <1> vga_mode_04h:      ; 320*200 graphics, 4 colors, CGA
00005FCE 2818080008          <1>       db     40, 24, 8, 00h, 08h   ; tw, th-1, ch, slength
00005FD3 09030002            <1>       db     09h, 03h, 00h, 02h ; sequ regs
00005FD7 63                  <1>       db     63h    ; misc reg
00005FD8 2D2728902B80BF1F    <1>       db     2Dh, 27h, 28h, 90h, 2Bh, 80h, 0BFh, 1Fh
00005FE0 00C1000000000000    <1>       db     00h, 0C1h, 00h, 00h, 00h, 00h, 00h, 00h
00005FE8 9C8E8F140096B9A2    <1>       db     9Ch, 8Eh, 8Fh, 14h, 00h, 96h, 0B9h, 0A2h
00005FF0 FF                  <1>       db     0FFh   ; crtc_regs
00005FF1 0013151702040607    <1>       db     00h, 13h, 15h, 17h, 02h, 04h, 06h, 07h
00005FF9 1011121314151617    <1>       db     10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
00006001 01000300            <1>       db     01h, 00h, 03h, 00h ; actl regs
00006005 0000000000300F0FFF  <1>       db     00h, 00h, 00h, 00h, 00h, 30h, 0Fh, 0Fh, 0FFh ; grdc regs
                             <1> vga_mode_06h:      ; 640*200 graphics, 2 colors, CGA
0000600E 5018080010          <1>       db     80, 24, 8, 00h, 10h   ;    tw, th-1, ch, slength
00006013 01010006            <1>       db     01h, 01h, 00h, 06h ; sequ regs
00006017 63                  <1>       db     63h    ; misc reg
00006018 5F4F50825480BF1F    <1>       db     5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
```

```
00006020 00C1000000000000    <1>       db     00h, 0C1h, 00h, 00h, 00h, 00h, 00h, 00h
00006028 9C8E8F280096B9C2    <1>       db     9Ch, 8Eh, 8Fh, 28h, 00h, 96h, 0B9h, 0C2h
00006030 FF                  <1>       db     0FFh   ; crtc regs
00006031 0017171717171717    <1>       db     00h, 17h, 17h, 17h, 17h, 17h, 17h, 17h
00006039 1717171717171717    <1>       db     17h, 17h, 17h, 17h, 17h, 17h, 17h, 17h
00006041 01000100            <1>       db     01h, 00h, 01, 00h  ; actl regs
00006045 0000000000000D0FFF  <1>       db     00h, 00h, 00h, 00h, 00h, 00h, 0Dh, 0Fh, 0FFh ; grdc regs
                             <1> vga_mode_13h:  ; mode 13h, 300*200, 256 colors, linear
0000604E 2818080000          <1>       db     40, 24, 8, 0, 0     ; tw, th-1, ch, slength (5)
00006053 010F000E            <1>       db     01h, 0Fh, 00h, 0Eh ; sequ regs (4)
00006057 63                  <1>       db     63h    ; misc reg (1)
00006058 5F4F50825480BF1F    <1>       db     5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
00006060 0041000000000000    <1>       db     00h, 41h, 00h, 00h, 00h, 00h, 00h, 00h
00006068 9C8E8F284096B9A3    <1>       db     9Ch, 8Eh, 8Fh, 28h, 40h, 96h, 0B9h, 0A3h
00006070 FF                  <1>       db     0FFh   ; crtc regs (25)
00006071 0001020304050607    <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
00006079 08090A0B0C0D0E0F    <1>       db     08h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh
00006081 41000F00            <1>       db     41h, 00h, 0Fh, 00h  ; actl regs (20)
00006085 000000000040050FFF  <1>       db     00h, 00h, 00h, 00h, 00h, 40h, 05h, 0Fh, 0FFh ; grdc regs (9)
                             <1> vga_mode_setl equ $ - vga_mode_13h  ; = 64
                             <1> vga_mode_F0h:  ; mode X ; 320*240, 256 colors, planar
0000608E 2818080000          <1>       db     40, 24, 8, 0, 0     ; tw, th-1, ch, slength
00006093 010F0006            <1>       db     01h, 0Fh, 00h, 06h ; sequ regs
00006097 E3                  <1>       db     0E3h   ; misc reg
00006098 5F4F508254800D3E    <1>       db     5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Dh, 3Eh
000060A0 0041000000000000    <1>       db     00h, 41h, 00h, 00h, 00h, 00h, 00h, 00h
000060A8 EAACDF2800E706E3    <1>       db     0EAh, 0ACh, 0DFh, 28h, 00h, 0E7h, 06h, 0E3h
000060B0 FF                  <1>       db     0FFh   ; crtc regs (25)
000060B1 0001020304050607    <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
000060B9 08090A0B0C0D0E0F    <1>       db     08h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh
000060C1 41000F00            <1>       db     41h, 00h, 0Fh, 00h  ; actl regs
000060C5 000000000040050FFF  <1>       db     00h, 00h, 00h, 00h, 00h, 40h, 05h, 0Fh, 0FFh ; grdc regs
                             <1> vga_mode_12h:  ; mode 12h, 640*480, 16 colors, planar
000060CE 501D100000          <1>       db     80, 29, 16, 0, 0 ; tw, th-1, ch, slength
000060D3 010F0006            <1>       db     01h, 0Fh, 00h, 06h ; sequ regs
000060D7 E3                  <1>       db     0E3h   ; misc reg
000060D8 5F4F508254800B3E    <1>       db     5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Bh, 3Eh
000060E0 0040000000000000    <1>       db     00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
000060E8 EA8CDF2800E704E3    <1>       db     0EAh, 8Ch, 0DFh, 28h, 00h, 0E7h, 04h, 0E3h
000060F0 FF                  <1>       db     0FFh   ; crtc regs
000060F1 0001020304051407    <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
000060F9 38393A3B3C3D3E3F    <1>       db     38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
00006101 01000F00            <1>       db     01h, 00h, 0Fh, 00h  ; actl regs
00006105 000000000000050FFF  <1>       db     00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
                             <1> vga_mode_6Ah:  ; mode 6Ah, 800*600, 16 colors, planar
0000610E 6424100000          <1>       db     100, 36, 16, 0, 0 ; tw, th-1, ch, slength
00006113 010F0006            <1>       db     01h, 0Fh, 00h, 06h ; sequ regs
00006117 E3                  <1>       db     0E3h   ; misc reg
00006118 7F6363836B1B72F0    <1>       db     7Fh, 63h, 63h, 83h, 6Bh, 1Bh, 72h, 0F0h
00006120 0060000000000000    <1>       db     00h, 60h, 00h, 00h, 00h, 00h, 00h, 00h
00006128 598D5732005773E3    <1>       db     59h, 8Dh, 57h, 32h, 00h, 57h, 73h, 0E3h
00006130 FF                  <1>       db     0FFh   ; crtc regs
00006131 0001020304051407    <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
00006139 38393A3B3C3D3E3F    <1>       db     38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
00006141 01000F00            <1>       db     01h, 00h, 0Fh, 00h  ; actl regs
00006145 000000000000050FFF  <1>       db     00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
                             <1> vga_mode_0Dh:  ; mode 0Dh, 320*200, 16 colors, planar
0000614E 2818080020          <1>       db     40, 24, 8, 0, 20h ; tw, th-1, ch, slength
00006153 090F0006            <1>       db     09h, 0Fh, 00h, 06h ; sequ regs
00006157 63                  <1>       db     63h    ; misc reg
00006158 2D2728902B80BF1F    <1>       db     2Dh, 27h, 28h, 90h, 2Bh, 80h, 0BFh, 1Fh
00006160 00C0000000000000    <1>       db     00h, 0C0h, 00h, 00h, 00h, 00h, 00h, 00h
00006168 9C8E8F140096B9E3    <1>       db     9Ch, 8Eh, 8Fh, 14h, 00h, 96h, 0B9h, 0E3h
00006170 FF                  <1>       db     0FFh   ; crtc regs
00006171 0001020304050607    <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
00006179 1011121314151617    <1>       db     10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
00006181 01000F00            <1>       db     01h, 00h, 0Fh, 00h  ; actl regs
00006185 000000000000050FFF  <1>       db     00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
                             <1> vga_mode_0Eh:  ; mode 0Eh, 640*200, 16 colors, planar
0000618E 5018080040          <1>       db     80, 24, 8, 0, 40h ; tw, th-1, ch, slength
00006193 010F0006            <1>       db     01h, 0Fh, 00h, 06h ; sequ regs
00006197 63                  <1>       db     63h    ; misc reg
00006198 5F4F50825480BF1F    <1>       db     5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
000061A0 00C0000000000000    <1>       db     00h, 0C0h, 00h, 00h, 00h, 00h, 00h, 00h
000061A8 9C8E8F280096B9E3    <1>       db     9Ch, 8Eh, 8Fh, 28h, 00h, 96h, 0B9h, 0E3h
000061B0 FF                  <1>       db     0FFh   ; crtc regs
000061B1 0001020304050607    <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
000061B9 1011121314151617    <1>       db     10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
000061C1 01000F00            <1>       db     01h, 00h, 0Fh, 00h  ; actl regs
000061C5 000000000000050FFF  <1>       db     00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
                             <1> vga_mode_10h: ; mode 10h, 640*350, 16 colors, planar
000061CE 50180E0080          <1>       db     80, 24, 14, 0, 80h ; tw, th-1, ch, slength
000061D3 010F0006            <1>       db     01h, 0Fh, 00h, 06h ; sequ regs
000061D7 A3                  <1>       db     0A3h   ; misc reg
000061D8 5F4F50825480BF1F    <1>       db     5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
000061E0 0040000000000000    <1>       db     00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
000061E8 83855D280F63BAE3    <1>       db     83h, 85h, 5Dh, 28h, 0Fh, 63h, 0BAh, 0E3h
000061F0 FF                  <1>       db     0FFh   ; crtc regs
000061F1 0001020304051407    <1>       db     00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
000061F9 38393A3B3C3D3E3F    <1>       db     38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
00006201 01000F00            <1>       db     01h, 00h, 0Fh, 00h  ; actl regs
00006205 000000000000050FFF  <1>       db     00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
                             <1> vga_mode_11h: ; mode 11h, 640*480, mono color, planar
0000620E 501D100000          <1>       db     80, 29, 16, 0, 0   ; tw, th-1, ch, slength
00006213 010F0006            <1>       db     01h, 0Fh, 00h, 06h ; sequ regs
00006217 E3                  <1>       db     0E3h   ; misc reg
00006218 5F4F508254800B3E    <1>       db     5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Bh, 3Eh
00006220 0040000000000000    <1>       db     00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
00006228 EA8CDF2800E704E3    <1>       db     0EAh, 8Ch, 0DFh, 28h, 00h, 0E7h, 04h, 0E3h
00006230 FF                  <1>       db     0FFh   ; crtc regs
00006231 003F003F003F003F    <1>       db     00h, 3Fh, 00h, 3Fh, 00h, 3Fh, 00h, 3Fh
00006239 003F003F003F003F    <1>       db     00h, 3Fh, 00h, 3Fh, 00h, 3Fh, 00h, 3Fh
00006241 01000F00            <1>       db     01h, 00h, 0Fh, 00h  ; actl regs
```

```
00006245 000000000000050FFF  <1>       db     00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
                             <1> end_of_vga_params:
                             <1> 
                             <1> ; /// End Of VIDEO DATA ///
                                 ;%include 'diskdata.s'    ; DISK (BIOS) DATA (initialized)
                                 ;;;
                                 
                                 Align 2
                                 
                                 %include 'sysdefs.s' ; 24/01/2015
                             <1> ; ****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - SYSTEM DEFINITIONS : sysdefs.s
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 31/12/2017
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 24/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                             <1> ; sysdefs.inc (14/11/2015)
                             <1> ; ****************************************************************************
                             <1> 
                             <1> ; Retro UNIX 386 v1 Kernel - SYSDEFS.INC
                             <1> ; Last Modification: 14/11/2015
                             <1> ;
                             <1> ; ///////// RETRO UNIX 386 V1 SYSTEM DEFINITIONS ///////////////
                             <1> ; (Modified from
                             <1> ;     Retro UNIX 8086 v1 system definitions in 'UNIX.ASM', 01/09/2014)
                             <1> ; ((UNIX.ASM (RETRO UNIX 8086 V1 Kernel), 11/03/2013 - 01/09/2014))
                             <1> ;     UNIX.ASM (MASM 6.11) --> SYSDEFS.INC (NASM 2.11)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ;
                             <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
                             <1> ; (Original) Source Code by Ken Thompson (1971-1972)
                             <1> ; <Bell Laboratories (17/3/1972)>
                             <1> ; <Preliminary Release of UNIX Implementation Document>
                             <1> ;
                             <1> ; ****************************************************************************
                             <1> 
                             <1> nproc        equ   16  ; number of processes
                             <1> nfiles       equ   50
                             <1> ntty  equ     8   ; 8+1 -> 8 (10/05/2013)
                             <1> nbuf  equ     4  ; 6 ;; 21/08/2015 - 'namei' buffer problem when nbuf > 4
                             <1>              ; NOTE: If fd0 super block buffer addres is beyond of the 1st
                             <1>              ; 32K, DMA r/w routine or someting else causes a jump to
                             <1>              ; kernel panic routine (in 'alloc' routine, in u5.s)
                             <1>              ; because of invalid buffer content (r/w error).
                             <1>              ; When all buffers are set before the end of the 1st 32k,
                             <1>              ; there is no problem!? (14/11/2015)
                             <1> 
                             <1> ;csgmnt      equ   2000h  ; 26/05/2013 (segment of process 1)
                             <1> ;core equ    0         ; 19/04/2013
                             <1> ;ecore       equ   32768 - 64  ; 04/06/2013 (24/05/2013)
                             <1>       ; (if total size of argument list and arguments is 128 bytes)
                             <1>       ; maximum executable file size = 32768-(64+40+128-6) = 32530 bytes
                             <1>       ; maximum stack size = 40 bytes (+6 bytes for 'IRET' at 32570)
                             <1>       ; initial value of user's stack pointer = 32768-64-128-2 = 32574
                             <1>       ;      (sp=32768-args_space-2 at the beginning of execution)
                             <1>       ; argument list offset = 32768-64-128 = 32576 (if it is 128 bytes)
                             <1>       ; 'u' structure offset (for the '/core' dump file) = 32704
                             <1>       ; '/core' dump file size = 32768 bytes
                             <1> 
                             <1> ; 08/03/2014
                             <1> ;sdsegmnt equ      6C0h  ; 256*16 bytes (swap data segment size for 16 processes)
                             <1> ; 19/04/2013 Retro UNIX 8086 v1 feaure only !
                             <1> ;;sdsegmnt equ     740h  ; swap data segment (for user structures and registers)
                             <1> 
                             <1> ; 30/08/2013
                             <1> time_count equ 4 ; 10 --> 4 01/02/2014
                             <1> 
                             <1> ; 05/02/2014
                             <1> ; process status
                             <1> ;SFREE       equ 0
                             <1> ;SRUN equ 1
                             <1> ;SWAIT       equ 2
                             <1> ;SZOMB       equ 3
                             <1> ;SSLEEP      equ 4 ; Retro UNIX 8086 V1 extension (for sleep and wakeup)
                             <1> 
                             <1> ; 09/03/2015
                             <1> userdata equ 80000h ; user structure data address for current user ; temporary
                             <1> swap_queue equ 90000h - 2000h ; swap queue address ; temporary
                             <1> swap_alloc_table equ 0D0000h  ;  swap allocation table address ; temporary
                             <1> 
                             <1> ; 17/09/2015
                             <1> ESPACE equ 48 ; [u.usp] (at 'sysent') - [u.sp] value for error return
                             <1> 
                             <1> ; 31/12/2017
                             <1> ; 19/02/2017
                             <1> ; 15/10/2016
                             <1> ; 20/05/2016
                             <1> ; 19/05/2016
                             <1> ; 18/05/2016
                             <1> ; 29/04/2016
                             <1> ; TRDOS 386 (TRDOS v2.0) system calls - temporary List
                             <1> ; 14/07/2013 - 21/09/2015 (Retro UNIX 8086 & 386 system calls)
                             <1> _ver  equ 0 ; Get TRDOS version (v2.0)
                             <1> _exit        equ 1
                             <1> _fork        equ 2
                             <1> _read        equ 3
                             <1> _write       equ 4
                             <1> _open equ 5
                             <1> _close       equ 6
```

```
<1> _wait       equ 7
<1> _creat      equ 8
<1> _rename     equ 9  ; TRDOS 386, Rename File (31/12/2017)
<1> _delete     equ 10 ; TRDOS 386, Delete File (29/12/2017)
<1> _exec equ 11
<1> _chdir      equ 12
<1> _time       equ 13 ; TRDOS 386, Get Sys Date&Time (30/12/2017)
<1> _mkdir      equ 14
<1> _chmod      equ 15 ; TRDOS 386, Change Attributes (30/12/2017)
<1> _rmdir      equ 16 ; TRDOS 386, Remove Directory (29/12/2017)
<1> _break      equ 17
<1> _drive      equ 18 ; TRDOS 386, Get/Set Current Drv (30/12/2017)
<1> _seek equ 19
<1> _tell       equ 20
<1> _mem  equ 21 ; TRDOS 386, Get Total&Free Mem (31/12/2017)
<1> _prompt     equ 22 ; TRDOS 386, Change Cmd Prompt (31/12/2017)
<1> _path equ 23 ; TRDOS 386, Get/Set Run Path (31/12/2017)
<1> _env  equ 24 ; TRDOS 386, Get/Set Env Vars (31/12/2017)
<1> _stime      equ 25 ; TRDOS 386, Set Sys Date&Time (30/12/2017)
<1> _quit equ 26
<1> _intr equ 27
<1> _dir  equ 28 ; TRDOS 386, Get Curr Drive&Dir (30/12/2017)
<1> _emt  equ 29
<1> _ldrvt      equ 30 ; TRDOS 386, Get Logical DOS DDT (30/12/2017)
<1> _video  equ 31 ; TRDOS 386 Video Functions (16/05/2016)
<1> _audio      equ 32 ; TRDOS 386 Video Functions (16/05/2016)
<1> _timer      equ 33 ; TRDOS 386 Timer Functions (18/05/2016)
<1> _sleep      equ 34 ; Retro UNIX 8086 v1 feature only !
<1> _msg  equ 35 ; Retro UNIX 386 v1 feature only !
<1> _geterr     equ 36 ; Retro UNIX 386 v1 feature only !
<1> _fpsave equ 37 ; TRDOS 386 FPU state option (28/02/2017)
<1> _pri  equ 38 ; change priority - TRDOS 386 (20/05/2016)
<1> _rele equ 39 ; TRDOS 386 (19/05/2016)
<1> _fff  equ 40 ; Find First File - TRDOS 386 (15/10/2016)
<1> _fnf  equ 41 ; Find Next File - TRDOS 386 (15/10/2016)
<1> _alloc      equ 42 ; Allocate memory - TRDOS 386 (19/02/2017)
<1>              ; TRDOS 386 (19/02/2017) DMA buff fuctions
<1> _dalloc equ 43 ; Deallocate mem - TRDOS 386 (19/02/2017)
<1> _calbac equ 44 ; Set IRQ callback - TRDOS 386 (20/02/2017)
<1> _dma  equ 45 ; DMA service - TRDOS 386 (20/08/2017)
<1>
<1> %macro sys 1-4
<1>     ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
<1>     ; 03/09/2015
<1>     ; 13/04/2015
<1>     ; Retro UNIX 386 v1 system call.
<1>     %if %0 >= 2
<1>         mov ebx, %2
<1>         %if %0 >= 3
<1>             mov ecx, %3
<1>             %if %0 = 4
<1>                mov edx, %4
<1>             %endif
<1>         %endif
<1>     %endif
<1>     mov eax, %1
<1>     ;int 30h
<1>     int 40h ; TRDOS 386 (TRDOS v2.0)
<1> %endmacro
<1>
<1> ; TRDOS 386 system calls, interrupt number
<1> ; 25/12/2016
<1> SYSCALL_INT_NUM   equ '40' ; '40h'
<1>
<1> ; 13/05/2015 - ERROR CODES
<1> ERR_FILE_NOT_OPEN  equ 10 ; 'file not open !' error
<1> ERR_FILE_ACCESS    equ 11 ; 'permission denied !' error
<1> ; 14/05/2015
<1> ERR_DIR_ACCESS     equ 11 ; 'permission denied !' error
<1> ERR_FILE_NOT_FOUND equ 12 ; 'file not found !' error
<1> ERR_TOO_MANY_FILES equ 13 ; 'too many open files !' error
<1> ERR_DIR_EXISTS     equ 14 ; 'directory already exists !' error
<1> ; 16/05/2015
<1> ERR_DRV_NOT_RDY    equ 15 ; 'drive not ready !' error
<1> ; 18/05/2015
<1> ERR_DEV_NOT_RDY    equ 15 ; 'device not ready !' error
<1> ERR_DEV_ACCESS     equ 11 ; 'permission denied !' error
<1> ERR_DEV_NOT_OPEN   equ 10 ; 'device not open !' error
<1> ; 07/06/2015
<1> ERR_FILE_EOF    equ 16 ; 'end of file !' error
<1> ERR_DEV_VOL_SIZE   equ 16 ; 'out of volume !' error
<1> ; 09/06/2015
<1> ERR_DRV_READ    equ 17 ; 'disk read error !'
<1> ERR_DRV_WRITE         equ 18 ; 'disk write error !'
<1> ; 16/06/2015
<1> ERR_NOT_DIR     equ 19 ; 'not a (valid) directory !' error
<1> ERR_FILE_SIZE         equ 20 ; 'file size error !'
<1> ; 22/06/2015
<1> ERR_NOT_SUPERUSER  equ 11 ; 'permission denied !' error
<1> ERR_NOT_OWNER      equ 11 ; 'permission denied !' error
<1> ERR_NOT_FILE       equ 11 ; 'permission denied !' error
<1> ; 23/06/2015
<1> ERR_FILE_EXISTS    equ 14 ; 'file already exists !' error
<1> ERR_DRV_NOT_SAME   equ 21 ; 'not same drive !' error
<1> ERR_DIR_NOT_FOUND  equ 12 ; 'directory not found !' error
<1> ERR_NOT_EXECUTABLE equ 22 ; 'not executable file !' error
<1> ; 27/06/2015
<1> ERR_INV_PARAMETER  equ 23 ; 'invalid parameter !' error
<1> ERR_INV_DEV_NAME   equ 24 ; 'invalid device name !' error
<1> ; 29/06/2015
<1> ERR_TIME_OUT    equ 25 ; 'time out !' error
<1> ERR_DEV_NOT_RESP   equ 25 ; 'device not responding !' error
<1> ; 10/10/2016
```

```
<1> ERR_INV_FILE_NAME  equ 26 ; 'invalid file name !' error
<1> ERR_INV_FLAGS         equ 23 ; 'invalid flags !' error
<1> ; For code compatibility with previous version of TRDOS (2011)
<1> ; (Temporary error codes for current TRDOS 386 -2016- version)
<1> ERR_NO_MORE_FILES  equ 12 ; 'no more files !' error
<1> ERR_PATH_NOT_FOUND equ  3 ; 'path not found !' error
<1>                      ; 'dir not found !' ; TRDOS 8086
<1> ERR_NOT_FOUND:        equ  2 ; 'file not found !' ; TRDOS 8086
<1> ERR_DISK_SPACE        equ 39 ; 'out of volume !' TRDOS 8086
<1>                      ; 'insufficient disk space !' ; 27h
<1> ERR_DISK_WRITE        equ 30 ; 'disk write protected !' ; 16/10/2016
<1> ERR_ACCESS_DENIED  equ  5 ; 'access denied !' ; TRDOS 8086
<1> ; 28/02/2017
<1> ERR_PERM_DENIED       equ 11 ; 'permission denied !' error
<1> ; 18/05/2016
<1> ERR_MISC        equ 27 ; miscellaneous/other errors
<1> ; 15/10/2016
<1> ; TRDOS 8086 -> TRDOS 386 (0Bh -> 28)
<1> ERR_INV_FORMAT        equ 28 ; 'invalid format !' error
<1> ; TRDOS 8086 -> TRDOS 386 (0Dh -> 29)
<1> ERR_INV_DATA    equ 29 ; 'invalid data !' error
<1> ; TRDOS 8086 -> TRDOS 386 (0Eh -> 20)
<1> ERR_ZERO_LENGTH       equ 20  ; 'zero length !' error
<1> ; TRDOS 8086 -> TRDOS 386 (15h -> 17, 1Dh -> 18, 1Eh -> 17)
<1> ERR_DRV_NR_READ       equ 17 ; 'drive not ready or read error !'
<1> ERR_DRV_NR_WRITE   equ 18 ; 'drive not ready or write error !'
<1> ; 15/10/2016
<1> ERR_INV_PATH_NAME  equ 19 ; 'bad path name !' error
<1> ERR_BAD_CMD_ARG       equ  1 ; 'bad command argument !' ; TRDOS 8086
<1> ERR_INV_FNUMBER       equ  1 ; 'invalid function number !' ; TRDOS 8086
<1> ERR_BIG_FILE    equ  8 ; 'big file & out of memory ! ; TRDOS 8086
<1> ERR_BIG_DATA    equ  8 ; 'big data & out of memory ! ; TRDOS 8086
<1> ERR_CLUSTER     equ 35 ; 'cluster not available !' ; TRDOS 8086
<1> ERR_OUT_OF_MEMORY  equ  4 ; 'out of memory !'
<1>                      ; 'insufficient memory !'
<1> ERR_P_VIOLATION       equ 6 ; 'protection violation !'
<1> ERR_PAGE_FAULT        equ 224 ;'page fault !' ;0E0h
<1> ERR_SWP_DISK_READ            equ 40
<1> ERR_SWP_DISK_NOT_PRESENT   equ 41
<1> ERR_SWP_SECTOR_NOT_PRESENT equ 42
<1> ERR_SWP_NO_FREE_SPACE      equ 43
<1> ERR_SWP_DISK_WRITE         equ 44
<1> ERR_SWP_NO_PAGE_TO_SWAP    equ 45
<1> ; 10/04/2017
<1> ERR_BUFFER      equ 46  ; 'buffer error !'
<1> ; 28/08/2017 (20/08/2017)
<1> ERR_DMA               equ -1  ; DMA buffer (allocation/misc.) error!
<1>
<1> ; 26/08/2015
<1> ; 24/07/2015
<1> ; 24/06/2015
<1> MAX_ARG_LEN     equ 256 ; max. length of sys exec arguments
<1> ; 01/07/2015
<1> MAX_MSG_LEN     equ 255 ; max. msg length for 'sysmsg'
<1> ;
<1> ; 06/10/2016
<1> OPENFILES       equ 10  ; max. number of open files (system)
<1> ; 07/10/2016
<1> ;NUMOFDEVICES         equ 20  ; max. num of available devices (sys)
<1>
    %include 'trdosk0.s' ; 04/01/2016
<1> ; *****************************************************************************
<1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - DEFINITIONS : trdosk0.s
<1> ; -----------------------------------------------------------------------------
<1> ; Last Update: 29/02/2016
<1> ; -----------------------------------------------------------------------------
<1> ; Beginning: 04/01/2016
<1> ; -----------------------------------------------------------------------------
<1> ; Assembler: NASM version 2.11 (trdos386.s)
<1> ; -----------------------------------------------------------------------------
<1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
<1> ; TRDOS2.ASM (09/11/2011)
<1> ; *****************************************************************************
<1> ; TRDOS2.ASM (c) 2004-2011 Erdogan TAN [ 17/01/2004 ] Last Update: 09/11/2011
<1> ;
<1> ; Masterboot / Partition Table at Beginning+1BEh
<1> ptBootable       equ 0
<1> ptBeginHead      equ 1
<1> ptBeginSector    equ 2
<1> ptBeginCylinder  equ 3
<1> ptFileSystemID   equ 4
<1> ptEndHead        equ 5
<1> ptEndSector      equ 6
<1> ptEndCylinder    equ 7
<1> ptStartSector    equ 8
<1> ptSectors        equ 12
<1>
<1> ; Boot Sector Parameters at 7C00h
<1> DataArea1     equ -4
<1> DataArea2     equ -2
<1> BootStart     equ 0h
<1> OemName       equ 03h
<1> BytesPerSec   equ 0Bh
<1> SecPerClust   equ 0Dh
<1> ResSectors    equ 0Eh
<1> FATs          equ 10h
<1> RootDirEnts   equ 11h
<1> Sectors       equ 13h
<1> Media         equ 15h
<1> FATSecs       equ 16h
<1> SecPerTrack   equ 18h
<1> Heads         equ 1Ah
<1> Hidden1       equ 1Ch
```

```
<1> Hidden2       equ 1Eh
<1> HugeSec1      equ 20h
<1> HugeSec2      equ 22h
<1> DriveNumber   equ 24h
<1> Reserved1     equ 25h
<1> bootsignature equ 26h
<1> VolumeID      equ 27h
<1> VolumeLabel   equ 2Bh
<1> FileSysType   equ 36h
<1> Reserved2     equ 3Eh                           ; Starting cluster of P2000
<1>
<1> ; FAT32 BPB Structure
<1> FAT32_FAT_Size equ 36
<1> FAT32_RootFClust equ 44
<1> FAT32_FSInfoSec equ 48
<1> FAT32_DrvNum equ 64
<1> FAT32_BootSig equ 66
<1> FAT32_VolID equ 67
<1> FAT32_VolLab equ 71
<1> FAT32_FilSysType equ 82
<1>
<1> ; BIOS Disk Parameters
<1> DPDiskNumber  equ 0h
<1> DPDType       equ 1h
<1> DPReturn      equ 2h
<1> DPHeads       equ 3h
<1> DPCylinders   equ 4h
<1> DPSecPerTrack equ 6h
<1> DPDisks       equ 7h
<1> DPTableOff    equ 8h
<1> DPTableSeg    equ 0Ah
<1> DPNumOfSecs   equ 0Ch
<1>
<1> ; BIOS INT 13h Extensions (LBA extensions)
<1> ; Just After DP Data (DPDiskNumber+)
<1> DAP_PacketSize equ 10h  ; If extensions present, this byte will be >=10h
<1> DAP_Reserved1 equ 11h   ; Reserved Byte
<1> DAP_NumOfBlocks equ 12h ; Value of this byte must be 0 to 127
<1> DAP_Reserved2 equ 13h   ; Reserved Byte
<1> DAP_Destination equ 14h ; Address of Transfer Buffer as SEGMENT:OFFSET
<1> DAP_LBA_Address equ 18h ; LBA=(C1*H0+H1)*S0+S1-1
<1>                         ; C1= Selected Cylinder Number
<1>                         ; H0= Number Of Heads (Maximum Head Number + 1)
<1>                         ; H1= Selected Head Number
<1>                         ; S0= Maximum Sector Number
<1>                         ; S1= Selected Sector Number
<1>                         ; QUAD WORD
<1> ; DAP_Flat_Destination equ 20h ; 64 bit address, if value in 4h is FFFF:FFFFh
<1>                              ; QUAD WORD (Also, value in 0h must be 18h)
<1>                              ; TR-DOS will not use 64 bit Flat Address
<1>
<1> ; INT 13h Function 48h "Get Enhanced Disk Drive Parameters"
<1> ; Just After DP Data (DPDiskNumber+)
<1> GetDParams_48h equ 20h ; Word. Data Length, must be 26 (1Ah) for short data.
<1> GDP_48h_InfoFlag equ 22h ; Word
<1> ; Bit 1 = 1 -> The geometry returned in bytes 4-15 is valid.
<1> GDP_48h_NumOfPCyls equ 24h ; Double Word. Number physical cylinders.
<1> GDP_48h_NumOfPHeads equ 28h ; Double Word. Number of physical heads.
<1> GDP_48h_NumOfPSpT equ 2Ch ; Double word. Num of physical sectors per track.
<1> GDP_48h_LBA_Sectors equ 30h ; 8 bytes. Number of physical/LBA sectors.
<1> GDP_48h_BytesPerSec equ 38h ; Word. Number of bytes in a sector.
<1>
<1> ; TR-DOS Standalone Program Extensions to the DiskParams Block
<1> ; Just After DP Data (DPDiskNumber+)
<1> TRDP_CurrentSector equ 3Ah  ; DX:AX (LBA)
<1> TRDP_SectorCount equ 3Eh    ; CX (or Counter)
<1>
<1>
<1> ; DOS Logical Disks
<1> LD_Name equ 0
<1> LD_DiskType equ 1
<1> LD_PhyDrvNo equ 2
<1> LD_FATType equ 3
<1> LD_FSType equ 4
<1> LD_LBAYes equ 5
<1> LD_BPB equ 6
<1> LD_FATBegin equ 96
<1> LD_ROOTBegin equ 100
<1> LD_DATABegin equ 104
<1> LD_StartSector equ 108
<1> LD_TotalSectors equ 112
<1> LD_FreeSectors equ 116
<1> LD_Clusters equ 120
<1> LD_PartitionEntry equ 124
<1> LD_DParamEntry equ 125
<1> LD_MediaChanged equ 126
<1> LD_CDirLevel equ 127
<1> LD_CurrentDirectory equ 128
<1>
<1> ; Singlix FS Extensions to DOS Logical Disks
<1> ; 03/01/2010 (LD_BPB compatibility for CHS r/w)
<1>
<1> LD_FS_Name equ 0
<1> LD_FS_DiskType equ 1
<1> LD_FS_PhyDrvNo equ 2
<1> LD_FS_FATType equ 3
<1> LD_FS_FSType equ 4
<1> LD_FS_LBAYes equ 5
<1> LD_FS_BPB equ 6
<1> LD_FS_MediaAttrib equ 6
<1> LD_FS_VersionMajor equ 7
<1> LD_FS_RootDirD equ 8
<1> LD_FS_MATLocation equ 12
```

```
<1> LD_FS_Reserved1 equ 16 ;1 reserved byte
<1> LD_FS_BytesPerSec equ 17 ; LD_BPB + 0Bh
<1> LD_FS_Reserved2 equ 19 ;2 reserved byte
<1> LD_FS_DATLocation equ 20
<1> LD_FS_DATSectors equ 24
<1> LD_FS_Reserved3 equ 28 ;3 reserved word
<1> LD_FS_SecPerTrack equ 30 ; LD_BPB + 18h
<1> LD_FS_NumHeads equ 32    ; LD_BPB + 1Ah
<1> LD_FS_UnDelDirD equ 34
<1> LD_FS_Reserved4 equ 38 ;4 reserved word
<1> LD_FS_VolumeSerial equ 40
<1> LD_FS_VolumeName equ 44
<1> LD_FS_BeginSector equ 108
<1> LD_FS_VolumeSize equ 112
<1> LD_FS_FreeSectors equ 116
<1> LD_FS_FirstFreeSector equ 120
<1> LD_FS_PartitionEntry equ 124
<1> LD_FS_DParamEntry equ 125
<1> LD_FS_MediaChanged equ 126
<1> LD_FS_CDirLevel equ 127
<1> LD_FS_CDIR_Converted equ 128
<1>
<1> ; Valid FAT Types
<1> FS_FAT12 equ 1
<1> FS_FAT16_CHS equ 2
<1> FS_FAT32_CHS equ 3
<1> FS_FAT16_LBA equ 4
<1> FS_FAT32_LBA equ 5
<1>
<1> ; Cursor Location
<1> CCCpointer equ  0450h   ; BIOS data, current cursor column
<1> ; FAT Clusters EOC sign
<1> FAT12EOC equ 0FFFh
<1> FAT16EOC equ 0FFFFh
<1> ;FAT32EOC equ 0FFFFFFFh ; It is not direct usable for 8086 code
<1> ; BAD Cluster
<1> FAT12BADC equ 0FF7h
<1> FAT16BADC equ 0FFF7h
<1> ;FAT32BADC equ 0FFFFFF7h ; It is not direct usable for 8086 code
<1> ; MS-DOS FAT16 FS (Maximum Possible) Last Cluster Number= 0FFF6h
<1>
<1> ; TRFS
<1>
<1> bs_FS_JmpBoot equ 0 ; jmp short bsBootCode
<1>                 ; db 0EBh, db 3Fh, db 90h
<1> bs_FS_Identifier equ 3  ; db 'FS', db 0
<1> bs_FS_BytesPerSec equ 6 ; dw 512
<1> bs_FS_MediaAttrib equ 8 ; db 3
<1> bs_FS_PartitionID equ 9 ; db 0A1h
<1> bs_FS_VersionMaj equ 10 ; db 01h
<1> bs_FS_VersionMin equ 11 ; db 0
<1> bs_FS_BeginSector equ 12   ; dd 0
<1> bs_FS_VolumeSize equ 16 ; dd 2880
<1> bs_FS_StartupFD equ 20 ; dd 0
<1> bs_FS_MATLocation equ 24 ; dd 1
<1> bs_FS_RootDirD equ 28 ; dd 8
<1> bs_FS_SystemConfFD equ 32 ; dd 0
<1> bs_FS_SwapFD equ 36 ; dd 0
<1> bs_FS_UnDelDirD equ 40 ; dd 0
<1> bs_FS_DriveNumber equ 44 ; db 0
<1> bs_FS_LBA_Ready equ 45 ; db 0
<1> bs_FS_MagicWord equ 46
<1> bs_FS_SecPerTrack equ 46 ; db 0A1h
<1> bs_FS_Heads equ 47 ; db 01h
<1> bs_FS_OperationSys equ 48 ; db "TR-SINGLIX v1.0b"
<1> bs_FS_Terminator equ 64 ; db 0
<1> bs_FS_BootCode equ 65
<1>
<1> FS_MAT_DATLocation equ 12
<1> FS_MAT_DATScount equ 16
<1> FS_MAT_FreeSectors equ 20
<1> FS_MAT_FirstFreeSector equ 24
<1> FS_RDT_VolumeSerialNo equ 28
<1> FS_RDT_VolumeName equ 64
<1>
<1> ; FAT12 + FAT16 + FAT32
<1> BS_JmpBoot equ 0
<1> BS_OEMName equ 3
<1> BPB_BytsPerSec equ 11
<1> BPB_SecPerClust equ 13
<1> BPB_RsvdSecCnt equ 14
<1> BPB_NumFATs equ 16
<1> BPB_RootEntCnt equ 17
<1> BPB_TotalSec16 equ 19
<1> BPB_Media equ 21
<1> BPB_FATSz16 equ 22
<1> BPB_SecPerTrk equ 24
<1> BPB_NumHeads equ 26
<1> BPB_HiddSec equ 28
<1> BPB_TotalSec32 equ 32
<1>
<1> ; FAT12 and FAT16 only
<1> BS_DrvNum equ 36
<1> BS_Reserved1 equ 37
<1> BS_BootSig equ 38
<1> BS_VolID equ 39
<1> BS_VolLab equ 43
<1> BS_FilSysType equ 54 ; 8 bytes
<1> BS_BootCode equ 62
<1>
<1> ; FAT32 only
<1> BPB_FATSz32 equ 36 ; FAT32, 4 bytes
<1> BPB_ExtFlags equ 40 ; FAT32, 2 bytes
```

```
                             <1> BPB_FSVer equ 42 ; FAT32, 2 bytes
                             <1> BPB_RootClus equ 44 ; FAT32, 4 bytes
                             <1> BPB_FSInfo equ 48 ; FAT 32, 2 bytes
                             <1> BPB_BkBootSec equ 50 ; FAT32, 2 bytes
                             <1> BPB_Reserved equ 52 ; FAT32, 12 bytes
                             <1> BS_FAT32_DrvNum equ 64 ; FAT32, 1 byte
                             <1> BS_FAT32_Reserved1 equ 65 ; FAT32, 1 byte
                             <1> BS_FAT32_BootSig equ 66 ; FAT32, 1 byte
                             <1> BS_FAT32_VolID equ 67 ; FAT32, 4 bytes
                             <1> BS_FAT32_VolLab equ 71 ; FAT32, 11 bytes
                             <1> BS_FAT32_FilSysType equ 82 ; FAT32, 8 bytes
                             <1> BS_FAT32_BootCode equ 90
                             <1> 
                             <1> ; 29/02/2016
                             <1> ;(FAT32 Free Cluster Count & First Free Cluster values)
                             <1> ;[BPB_Reserved] = Free Cluster Count (offset 52)
                             <1> ;[BPB_Reserved+4] = First Free Cluster (offset 56)
                             <1> 
                             <1> BS_Validation equ 510
                             <1> 
                             <1> ; 15/02/2016
                             <1> ; FILE.ASM - 09/10/2011
                             <1> ; Directory Entry Structure
                             <1> ; 29/10/2009 (According to Microsoft FAT32 File System Specification)
                             <1> DirEntry_Name equ 0
                             <1> DirEntry_Attr equ 11
                             <1> DirEntry_NTRes equ 12
                             <1> DirEntry_CrtTimeTenth equ 13
                             <1> DirEntry_CrtTime equ 14
                             <1> DirEntry_CrtDate equ 16
                             <1> DirEntry_LastAccDate equ 18
                             <1> DirEntry_FstClusHI equ 20
                             <1> DirEntry_WrtTime equ 22
                             <1> DirEntry_WrtDate equ 24
                             <1> DirEntry_FstClusLO equ 26
                             <1> DirEntry_FileSize equ 28
                                 %include 'trdosk1.s' ; 04/01/2016
                             <1> ; ****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - SYS INIT : trdosk1.s
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 31/12/2017
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 04/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
                             <1> ; TRDOS2.ASM (09/11/2011)
                             <1> ; ****************************************************************************
                             <1> ; TRDOS2.ASM (c) 2004-2011 Erdogan TAN [ 17/01/2004 ] Last Update: 09/11/2011
                             <1> ;
                             <1> 
                             <1> sys_init:
                             <1>       ; 23/01/2017
                             <1>       ; 07/05/2016
                             <1>       ; 02/05/2016
                             <1>       ; 24/04/2016
                             <1>       ; 14/04/2016
                             <1>       ; 13/04/2016
                             <1>       ; 30/03/2016
                             <1>       ; 24/01/2016
                             <1>       ; 06/01/2016
                             <1>       ; 04/01/2016
                             <1> 
                             <1>       ; 23/01/2017 - reset timer frequency (to 18.2Hz)
0000624E B036                <1>       mov    al, 00110110b ; 36h
00006250 E643                <1>       out    43h, al
00006252 31C0                <1>       xor    eax, eax  ; sub     al, al ; 0
00006254 E640                <1>       out    40h, al ; LB
00006256 E640                <1>       out    40h, al ; HB
                             <1>       ;
                             <1>       ; 30/03/2016
                             <1>       ; Clear Logical DOS Disk Description Tables Area
                             <1>       ;xor   eax, eax
00006258 BF00010900          <1>       mov    edi, Logical_DOSDisks
0000625D B980060000          <1>       mov    ecx, 6656/4 ; 26*256 = 6656 bytes
00006262 F3AB                <1>       rep    stosd ; 1664 times 4 bytes
                             <1> 
00006264 B83F3A2F00          <1>       mov    eax, '?:/'
00006269 A3[FF580100]        <1>       mov    [Current_Dir_Drv], eax
                             <1> 
                             <1>       ; Logical DRV INIT (only for hard disks)
0000626E E803040000          <1>       call   ldrv_init  ; trdosk2.s
                             <1> 
                             <1>       ; When floppy_drv_init call is disabled
                             <1>       ; media changed sign is needed
                             <1>       ; for proper drive initialization
                             <1> 
00006273 BE00010900          <1>       mov    esi, Logical_DOSDisks
00006278 B001                <1>       mov    al, 1 ; Initialization sign (invalid_fd_parameter)
0000627A 83C67E              <1>       add    esi, LD_MediaChanged ; Media Change Status = 1 (init needed)
0000627D 8806                <1>       mov    [esi], al ; A:
0000627F 81C600010000        <1>       add    esi, 100h
00006285 8806                <1>       mov    [esi], al ; B:
                             <1> 
                             <1> _current_drive_bootdisk:
00006287 8A15[F25C0000]      <1>       mov    dl, [boot_drv] ; physical drive number
0000628D 80FAFF              <1>       cmp    dl, 0FFh
00006290 740A                <1>       je     short _last_dos_diskno_check
                             <1> _boot_drive_check:
00006292 80FA80              <1>       cmp    dl, 80h
00006295 7218                <1>       jb     short _current_drive_a
00006297 80EA7E              <1>       sub    dl, 7Eh ; C = 2 , D = 3
```

```
0000629A EB13                <1>       jmp    short _current_drive_a
                             <1> 
                             <1> _last_dos_diskno_check:
0000629C 8A15[D20C0100]      <1>       mov    dl, [Last_DOS_DiskNo]
000062A2 80FA02              <1>       cmp    dl, 2
000062A5 7706                <1>       ja     short _current_drive_c
000062A7 7406                <1>       je     short _current_drive_a
000062A9 30D2                <1>       xor    dl, dl ; A:
000062AB EB02                <1>       jmp    short _current_drive_a
                             <1> 
                             <1> _current_drive_c:
000062AD B202                <1>       mov    dl, 2 ; C:
                             <1> 
                             <1> _current_drive_a:
000062AF 8815[F35C0000]      <1>       mov    [drv], dl
000062B5 BE[D40C0100]        <1>         mov     esi, msg_CRLF_temp
000062BA E89E000000          <1>       call   print_msg
                             <1> 
000062BF 8A15[F35C0000]      <1>       mov    dl, [drv]
000062C5 E8F60B0000          <1>       call   change_current_drive
000062CA 730C                <1>       jnc    short _start_mainprog
                             <1> 
                             <1> _drv_not_ready_error:
000062CC BE[8F0F0100]        <1>       mov    esi, msgl_drv_not_ready
000062D1 E887000000          <1>       call   print_msg
000062D6 EB63                <1>         jmp     _end_of_mainprog
                             <1> 
                             <1> _start_mainprog:
                             <1>       ; 07/01/2017
                             <1>       ; 07/05/2016
                             <1>       ; 02/05/2016
                             <1>       ; 24/04/2016
                             <1>       ; Retro UNIX 386 v1, 'sys_init' (u0.s)
                             <1>       ; 23/06/2015
                             <1> 
                             <1>       ; 02/05/2016
                             <1>       ; 24/04/2016
000062D8 66B80100            <1>       mov    ax, 1
000062DC A2[B3030300]        <1>       mov    [u.uno], al
000062E1 66A3[4E030300]      <1>       mov    [mpid], ax
000062E7 66A3[20000300]      <1>       mov    [p.pid], ax
000062ED A2[B0000300]        <1>       mov    [p.stat], al
000062F2 C605[A8030300]04    <1>       mov    byte [u.quant], time_count  ; 07/01/2017
                             <1>       ;
000062F9 A1[38580100]        <1>       mov    eax, [k_page_dir]
000062FE A3[B8030300]        <1>       mov    [u.pgdir], eax ; reset
                             <1>       ;
00006303 E872E8FFFF          <1>       call   allocate_page
00006308 0F82A3000000        <1>       jc     panic
0000630E A3[B4030300]        <1>       mov    [u.upage], eax ; user structure page
00006313 A3[C0000300]        <1>       mov    [p.upage], eax
00006318 E8D7E8FFFF          <1>       call   clear_page
                             <1>       ;
                             <1>       ; 24/08/2015
0000631D FE0D[5B030300]      <1>       dec    byte [sysflg] ; FFh = ready for system call
                             <1>                           ; 0 = executing a system call
                             <1>       ; 13/04/2016
                             <1>       ; Clear Environment Variables Page/Area
00006323 BF00300900          <1>       mov    edi, Env_Page ; 93000h
00006328 B980000000          <1>       mov    ecx, Env_Page_Size / 4    ; 512/4  (4096/4)

0000632D 31C0                <1>       xor    eax, eax
0000632F F3AB                <1>       rep    stosd
                             <1> 
                             <1>       ; 14/04/2016
00006331 E8E1340000          <1>       call   mainprog_startup_configuration
                             <1> 
00006336 E8C60C0000          <1>         call    dos_prompt
                             <1> 
                             <1> _end_of_mainprog:
0000633B BE[D40C0100]        <1>         mov     esi, msg_CRLF_temp
00006340 E818000000          <1>       call   print_msg
00006345 BE[DA0C0100]        <1>       mov    esi, mainprog_Version
0000634A E80E000000          <1>       call   print_msg
                             <1>       ; 24/01/2016
0000634F 28E4                <1>       sub    ah, ah
00006351 E8C0A8FFFF          <1>       call   int16h ; call getch
00006356 E9A0ADFFFF          <1>       jmp    cpu_reset
                             <1> 
0000635B EBFE                <1> infinitiveloop: jmp short infinitiveloop
                             <1> 
                             <1> print_msg:
                             <1>       ; 13/05/2016
                             <1>       ; 04/01/2016
                             <1>       ; 01/07/2015
                             <1>       ; 13/03/2015 (Retro UNIX 386 v1)
                             <1>       ; 07/03/2014 (Retro UNIX 8086 v1)
                             <1>       ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
                             <1>       ;
0000635D 8A3D[66580100]      <1>       mov    bh, [ACTIVE_PAGE] ; 04/01/2016 (ptty)
                             <1>       ;mov   bl, 07h ; Black background, light gray forecolor
                             <1> 
00006363 AC                  <1>       lodsb
                             <1> pmsg1:
00006364 56                  <1>       push   esi
                             <1>       ;mov   bh, [ACTIVE_PAGE] ; 04/01/2016 (ptty)
00006365 B307                <1>       mov    bl, 07h ; Black background, light gray forecolor
00006367 E846B9FFFF          <1>       call   _write_tty
0000636C 5E                  <1>       pop    esi
0000636D AC                  <1>       lodsb
0000636E 20C0                <1>       and    al, al
00006370 75F2                <1>       jnz    short pmsg1
00006372 C3                  <1>       retn
```

```
                             <1>
                             <1> clear_screen:
                             <1>       ; 13/05/2016
                             <1>       ; 30/01/2016
                             <1>       ; 24/01/2016
                             <1>       ; 04/01/2016
00006373 0FB61D[66580100]    <1>       movzx  ebx, byte [ACTIVE_PAGE] ; video page number (0 to 7)
0000637A 8AA3[D35E0000]      <1>       mov    ah, [ebx+vmode] ; default = 03h (80x25 text)
00006380 80FC04              <1>       cmp    ah, 4
00006383 7205                <1>       jb     short cls1
00006385 80FC07              <1>       cmp    ah, 7
00006388 7526                <1>       jne    short vga_clear
                             <1> cls1:
                             <1>       ;mov   bh, bl
                             <1>       ;mov   bl, 7
0000638A 3A25[C25E0000]      <1>       cmp    ah, [CRT_MODE] ; current video mode ?
                             <1>       ;je    short cls2 ; yes (current video mode = 3)
                             <1>       ;;call set_mode_3 ; set video mode to 3 (& clear screen)
                             <1>       ;;retn
                             <1>       ;jmp   set_mode_3
00006390 0F8526B9FFFF        <1>       jne    set_mode_3
                             <1> cls2:
00006396 88DF                <1>       mov    bh, bl ; video page (0 to 7)
00006398 B307                <1>       mov    bl, 07h ; attribute to be used on blanked line
0000639A 28C0                <1>       sub    al, al ; 0 =  entire window
0000639C 6631C9              <1>       xor    cx, cx
0000639F 66BA4F18            <1>       mov    dx, 184Fh
000063A3 E862B6FFFF          <1>       call   _scroll_up ; 24/01/2016
                             <1>       ;
                             <1>       ;mov   bh, [ACTIVE_PAGE] ; video page number (0 to 7)
000063A8 6631D2              <1>       xor    dx, dx
000063AB E898B9FFFF          <1>       call   _set_cpos ; 24/01/2016
                             <1>       ;retn
                             <1> vga_clear:
000063B0 C3                  <1>       retn
                             <1>
                             <1> panic:
                             <1>       ; 13/05/2016 (TRDOS 386 = TRDOS v2)
                             <1>       ; 13/03/2015 (Retro UNIX 386 v1)
                             <1>       ; 07/03/2014 (Retro UNIX 8086 v1)
000063B1 BE[72190100]        <1>       mov    esi, panic_msg
000063B6 E8A2FFFFFF          <1>       call   print_msg
                             <1> key_to_reboot:
                             <1>         ; 24/01/2016
000063BB 28E4                <1>         sub     ah, ah
000063BD E854A8FFFF          <1>         call    int16h ; call   getch
                             <1>         ; wait for a character from the current tty
                             <1>       ;
000063C2 B00A                <1>       mov    al, 0Ah
000063C4 8A3D[66580100]      <1>       mov    bh, [ptty] ; [ACTIVE_PAGE]
000063CA B307                <1>       mov    bl, 07h ; Black background,
                             <1>                    ; light gray forecolor
000063CC E8E1B8FFFF          <1>       call   _write_tty
000063D1 E925ADFFFF          <1>       jmp    cpu_reset
                             <1>
                             <1> ctrlbrk:
                             <1>       ; 12/11/2015
                             <1>       ; 13/03/2015 (Retro UNIX 386 v1)
                             <1>       ; 06/12/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; INT 1Bh (control+break) handler
                             <1>       ;
                             <1>             ; Retro Unix 8086 v1 feature only!
                             <1>             ;
000063D6 66833D[AA030300]00  <1>       cmp    word [u.intr], 0
000063DE 7645                <1>       jna    short cbrk4
                             <1> cbrk0:
                             <1>       ; 12/11/2015
                             <1>       ; 06/12/2013
000063E0 66833D[AC030300]00  <1>       cmp    word [u.quit], 0
000063E8 743B                <1>       jz     short cbrk4
                             <1>       ;
                             <1>       ; 20/09/2013
000063EA 6650                <1>       push   ax
000063EC A0[66580100]        <1>       mov    al, [ptty]
                             <1>       ;
                             <1>       ; 12/11/2015
                             <1>       ;
                             <1>       ; ctrl+break (EOT, CTRL+D) from serial port
                             <1>       ; or ctrl+break from console (pseudo) tty
                             <1>       ; (!redirection!)
                             <1>       ;
000063F1 3C08                <1>       cmp    al, 8 ; serial port tty nums > 7
000063F3 7211                <1>         jb      short cbrk1 ; console (pseudo) tty
                             <1>       ;
                             <1>       ; Serial port interrupt handler sets [ptty]
                             <1>       ; to the port's tty number (as temporary).
                             <1>       ;
                             <1>       ; If active process is using a stdin or
                             <1>       ; stdout redirection (by the shell),
                             <1>         ; console tty keyboard must be available
                             <1>       ; to terminate running process,
                             <1>       ; in order to prevent a deadlock.
                             <1>       ;
000063F5 52                  <1>       push   edx
000063F6 0FB615[B3030300]    <1>       movzx  edx, byte [u.uno]
000063FD 3A82[7F000300]      <1>       cmp     al, [edx+p.ttyc-1] ; console tty (rw)
00006403 5A                  <1>       pop    edx
00006404 7412                <1>       je     short cbrk2
                             <1> cbrk1:
00006406 FEC0                <1>       inc    al  ; [u.ttyp] : 1 based tty number
                             <1>       ; 06/12/2013
00006408 3A05[94030300]      <1>       cmp    al, [u.ttyp] ; recent open tty (r)
```

```
0000640E 7408                <1>       je     short cbrk2
00006410 3A05[95030300]      <1>         cmp     al, [u.ttyp+1] ; recent open tty (w)
00006416 750B                <1>       jne    short cbrk3
                             <1> cbrk2:
                             <1>       ;; 06/12/2013
                             <1>       ;mov   ax, [u.quit]
                             <1>       ;and   ax, ax
                             <1>       ;jz    short cbrk3
                             <1>       ;
00006418 6631C0              <1>       xor    ax, ax ; 0
0000641B 6648                <1>       dec    ax
                             <1>       ; 0FFFFh = 'ctrl+brk' keystroke
0000641D 66A3[AC030300]      <1>       mov    [u.quit], ax
                             <1> cbrk3:
00006423 6658                <1>       pop    ax
                             <1> cbrk4:
00006425 C3                  <1>       retn
                             <1>
                             <1>
                             <1> ; 31/12/2017
                             <1> ; TRDOS 386 - 30/12/2017
                             <1> %define get_rtc_date RTC_40
                             <1> %define get_rtc_time RTC_20
                             <1> %define     set_rtc_date RTC_50
                             <1> %define set_rtc_time RTC_30
                             <1> get_rtc_date_time:
                             <1> ; Retro UNIX 8086 v1 - UNIX.ASM (01/09/2014)
                             <1> ;epoch:
                             <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)
                             <1>       ; 09/04/2013 (Retro UNIX 8086 v1 - UNIX.ASM)
                             <1>       ; 'epoch' procedure prototype:
                             <1>       ;                 UNIXCOPY.ASM, 10/03/2013
                             <1>       ; 14/11/2012
                             <1>       ; unixboot.asm (boot file configuration)
                             <1>       ; version of "epoch" procedure in "unixproc.asm"
                             <1>       ; 21/7/2012
                             <1>       ; 15/7/2012
                             <1>       ; 14/7/2012
                             <1>       ; Erdogan Tan - RETRO UNIX v0.1
                             <1>       ; compute current date and time as UNIX Epoch/Time
                             <1>       ; UNIX Epoch: seconds since 1/1/1970 00:00:00
                             <1>       ;
                             <1>         ;  ((Modified registers: EAX, EDX, ECX, EBX))
                             <1>       ;
                             <1>
00006426 E8C3F5FFFF          <1>       call   get_rtc_time        ; Return Current Time
0000642B 86E9                <1>         xchg       ch,cl
0000642D 66890D[30550100]    <1>         mov  [hour], cx
00006434 86F2                <1>         xchg       dh,dl
00006436 668915[34550100]    <1>         mov  [second], dx
                             <1>       ;
0000643D E81DF6FFFF          <1>         call       get_rtc_date        ; Return Current Date
00006442 86E9                <1>         xchg       ch,cl
00006444 66890D[2A550100]    <1>         mov  [year], cx
0000644B 86F2                <1>         xchg       dh,dl
0000644D 668915[2C550100]    <1>         mov  [month], dx
                             <1>       ;
00006454 66B93030            <1>       mov    cx, 3030h
                             <1>       ;
00006458 A0[30550100]        <1>       mov    al, [hour] ; Hour
                             <1>             ; AL <= BCD number)
0000645D D410                <1>         db   0D4h,10h            ; Undocumented inst. AAM
                             <1>                                  ; AH = AL / 10h
                             <1>                                  ; AL = AL MOD 10h
0000645F D50A                <1>         aad  ; AX= AH*10+AL
00006461 A2[30550100]        <1>       mov    [hour], al
00006466 A0[31550100]        <1>       mov    al, [hour+1] ; Minute
                             <1>             ; AL <= BCD number)
0000646B D410                <1>         db   0D4h,10h            ; Undocumented inst. AAM
                             <1>                                  ; AH = AL / 10h
                             <1>                                  ; AL = AL MOD 10h
0000646D D50A                <1>         aad  ; AX= AH*10+AL
0000646F A2[32550100]        <1>       mov    [minute], al
00006474 A0[34550100]        <1>       mov    al, [second] ; Second
                             <1>             ; AL <= BCD number)
00006479 D410                <1>         db   0D4h,10h            ; Undocumented inst. AAM
                             <1>                                  ; AH = AL / 10h
                             <1>                                  ; AL = AL MOD 10h
0000647B D50A                <1>         aad  ; AX= AH*10+AL
0000647D A2[34550100]        <1>       mov    [second], al
00006482 66A1[2A550100]      <1>       mov    ax, [year] ; Year (century)
00006488 6650                <1>         push       ax
                             <1>             ; AL <= BCD number)
0000648A D410                <1>         db   0D4h,10h            ; Undocumented inst. AAM
                             <1>                                  ; AH = AL / 10h
                             <1>                                  ; AL = AL MOD 10h
0000648C D50A                <1>         aad  ; AX= AH*10+AL
0000648E B464                <1>       mov    ah, 100
00006490 F6E4                <1>       mul    ah
00006492 66A3[2A550100]      <1>       mov    [year], ax
00006498 6658                <1>       pop    ax
0000649A 88E0                <1>       mov    al, ah
                             <1>             ; AL <= BCD number)
0000649C D410                <1>         db   0D4h,10h            ; Undocumented inst. AAM
                             <1>                                  ; AH = AL / 10h
                             <1>                                  ; AL = AL MOD 10h
0000649E D50A                <1>         aad  ; AX= AH*10+AL
000064A0 660105[2A550100]    <1>       add    [year], ax
000064A7 A0[2C550100]        <1>       mov    al, [month] ; Month
                             <1>             ; AL <= BCD number)
000064AC D410                <1>         db   0D4h,10h            ; Undocumented inst. AAM
                             <1>                                  ; AH = AL / 10h
                             <1>                                 ; AL = AL MOD 10h
000064AE D50A                <1>         aad ; AX= AH*10+AL
000064B0 A2[2C550100]        <1>       mov    [month], al
000064B5 A0[2D550100]        <1>         mov     al, [month+1]          ; Day
                             <1>             ; AL <= BCD number)
000064BA D410                <1>         db  0D4h,10h            ; Undocumented inst. AAM
                             <1>                                 ; AH = AL / 10h
                             <1>                                 ; AL = AL MOD 10h
000064BC D50A                <1>         aad ; AX= AH*10+AL
000064BE A2[2E550100]        <1>         mov     [day], al
                             <1> 
000064C3 C3                  <1>       retn   ; 30/12/2017
                             <1> 
                             <1> epoch:
000064C4 E85DFFFFFF          <1>       call   get_rtc_date_time ; TRDOS 386 - 30/12/2017
                             <1> 
                             <1> convert_to_epoch:
                             <1>       ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit modification)
                             <1>       ; 09/04/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; ((Modified registers: EAX, EDX, EBX))
                             <1>       ;
                             <1>       ; Derived from DALLAS Semiconductor
                             <1>       ; Application Note 31 (DS1602/DS1603)
                             <1>       ; 6 May 1998
000064C9 29C0                <1>       sub    eax, eax
000064CB 66A1[2A550100]      <1>       mov    ax, [year]
000064D1 662DB207            <1>       sub    ax, 1970
000064D5 BA6D010000          <1>       mov    edx, 365
000064DA F7E2                <1>       mul    edx
000064DC 31DB                <1>       xor    ebx, ebx
000064DE 8A1D[2C550100]      <1>       mov    bl, [month]
000064E4 FECB                <1>       dec    bl
000064E6 D0E3                <1>       shl    bl, 1
                             <1>       ;sub   edx, edx
000064E8 668B93[36550100]    <1>       mov    dx, [EBX+DMonth]
000064EF 8A1D[2E550100]      <1>         mov     bl, [day]
000064F5 FECB                <1>       dec    bl
000064F7 01D0                <1>       add    eax, edx
000064F9 01D8                <1>       add    eax, ebx
                             <1>                    ; EAX = days since 1/1/1970
000064FB 668B15[2A550100]    <1>       mov    dx, [year]
00006502 6681EAB107          <1>       sub    dx, 1969
00006507 66D1EA              <1>       shr    dx, 1
0000650A 66D1EA              <1>       shr    dx, 1
                             <1>             ; (year-1969)/4
0000650D 01D0                <1>       add    eax, edx
                             <1>                    ; + leap days since 1/1/1970
0000650F 803D[2C550100]02    <1>       cmp    byte [month], 2     ; if past february
00006516 7610                <1>       jna    short cte1
00006518 668B15[2A550100]    <1>       mov    dx, [year]
0000651F 6683E203            <1>       and    dx, 3 ; year mod 4
00006523 7503                <1>       jnz    short cte1
                             <1>                    ; and if leap year
00006525 83C001              <1>       add    eax, 1       ; add this year's leap day (february 29)
                             <1> cte1:                    ; compute seconds since 1/1/1970
00006528 BA18000000          <1>       mov    edx, 24
0000652D F7E2                <1>       mul    edx
0000652F 8A15[30550100]      <1>       mov    dl, [hour]
00006535 01D0                <1>       add    eax, edx
                             <1>             ; EAX = hours since 1/1/1970 00:00:00
                             <1>       ;mov   ebx, 60
00006537 B33C                <1>       mov    bl, 60
00006539 F7E3                <1>       mul    ebx
0000653B 8A15[32550100]      <1>       mov    dl, [minute]
00006541 01D0                <1>       add    eax, edx
                             <1>             ; EAX = minutes since 1/1/1970 00:00:00
                             <1>       ;mov   ebx, 60
00006543 F7E3                <1>       mul    ebx
00006545 8A15[34550100]      <1>       mov    dl, [second]
0000654B 01D0                <1>       add    eax, edx
                             <1>             ; EAX -> seconds since 1/1/1970 00:00:00
0000654D C3                  <1>       retn
                             <1> 
                             <1> ;set_date_time:
                             <1> convert_from_epoch:
                             <1>       ; 31/12/2017
                             <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)
                             <1>       ; 20/06/2013 (Retro UNIX 8086 v1)
                             <1>       ; 'convert_from_epoch' procedure prototype:
                             <1>       ;                 UNIXCOPY.ASM, 10/03/2013
                             <1>       ;
                             <1>       ; ((Modified registers: EAX, EDX, ECX, EBX))
                             <1>       ;
                             <1>       ; Derived from DALLAS Semiconductor
                             <1>       ; Application Note 31 (DS1602/DS1603)
                             <1>       ; 6 May 1998
                             <1>       ;
                             <1>       ; INPUT:
                             <1>       ; EAX = Unix (Epoch) Time
                             <1>       ;
0000654E 31D2                <1>       xor    edx, edx
00006550 B93C000000          <1>       mov    ecx, 60
00006555 F7F1                <1>       div    ecx
                             <1>       ;mov   [imin], eax   ; whole minutes
                             <1>                      ; since 1/1/1970
00006557 668915[34550100]    <1>       mov    [second], dx  ; leftover seconds
0000655E 29D2                <1>       sub    edx, edx
00006560 F7F1                <1>       div    ecx
                             <1>       ;mov   [ihrs], eax   ; whole hours
                             <1>       ;                  ; since 1/1/1970
```

```
00006562 668915[32550100]    <1>       mov    [minute], dx  ; leftover minutes
00006569 31D2                <1>       xor    edx, edx
                             <1>       ;mov   cx, 24
0000656B B118                <1>       mov    cl, 24
0000656D F7F1                <1>       div    ecx
                             <1>       ;mov   [iday], ax   ; whole days
                             <1>                         ; since 1/1/1970
0000656F 668915[30550100]    <1>       mov    [hour], dx   ; leftover hours
00006576 05DB020000          <1>       add    eax, 365+366 ; whole day since
                             <1>                         ; 1/1/1968
                             <1>       ;mov   [iday], ax
0000657B 50                  <1>       push   eax
0000657C 29D2                <1>       sub    edx, edx
0000657E B9B5050000          <1>       mov    ecx, (4*365)+1 ; 4 years = 1461 days
00006583 F7F1                <1>       div    ecx
00006585 59                  <1>       pop    ecx
                             <1>       ;mov   [lday], ax   ; count of quadyrs (4 years)
00006586 6652                <1>       push   dx
                             <1>       ;mov   [qday], dx   ; days since quadyr began
00006588 6683FA3C            <1>       cmp    dx, 31 + 29  ; if past feb 29 then
0000658C F5                  <1>       cmc               ; add this quadyr's leap day
0000658D 83D000              <1>       adc    eax, 0     ; to # of qadyrs (leap days)
                             <1>       ;mov   [lday], ax   ; since 1968
                             <1>       ;mov   cx, [iday]
00006590 91                  <1>       xchg   ecx, eax     ; ECX = lday, EAX = iday
00006591 29C8                <1>       sub    eax, ecx     ; iday - lday
00006593 B96D010000          <1>       mov    ecx, 365
00006598 31D2                <1>       xor    edx, edx
                             <1>       ; EAX = iday-lday, EDX = 0
0000659A F7F1                <1>       div    ecx
                             <1>       ;mov   [iyrs], ax   ; whole years since 1968
                             <1>       ;jday = iday - (iyrs*365) - lday
                             <1>       ;mov [jday], dx      ; days since 1/1 of current year
                             <1>       ;add   eax, 1968
0000659C 6605B007            <1>       add    ax, 1968     ; compute year
000065A0 66A3[2A550100]      <1>       mov    [year], ax
000065A6 6689D1              <1>       mov    cx, dx
                             <1>       ;mov   dx, [qday]
000065A9 665A                <1>       pop    dx
000065AB 6681FA6D01          <1>       cmp    dx, 365           ; if qday <= 365 and qday >= 60
000065B0 7709                <1>       ja     short cfe1   ; jday = jday +1
000065B2 6683FA3C            <1>       cmp    dx, 60       ; if past 2/29 and leap year then
000065B6 F5                  <1>         cmc             ; add a leap day to the # of whole
000065B7 6683D100            <1>       adc    cx, 0        ; days since 1/1 of current year
                             <1> cfe1:
                             <1>       ;mov   [jday], cx
000065BB 66BB0C00            <1>       mov    bx, 12       ; estimate month
000065BF 66BA6E01            <1>       mov    dx, 366      ; mday, max. days since 1/1 is 365
000065C3 6683E003            <1>       and    ax, 11b      ; year mod 4 (and dx, 3)
                             <1> cfe2: ; Month calculation  ; 0 to 11  (11 to 0)
000065C7 6639D1              <1>       cmp    cx, dx       ; mday = # of days passed from 1/1
000065CA 731D                <1>       jnb    short cfe3
000065CC 664B                <1>       dec    bx           ; month = month - 1
000065CE 66D1E3              <1>       shl    bx, 1
000065D1 668B93[36550100]    <1>       mov    dx, [EBX+DMonth] ; # elapsed days at 1st of month
000065D8 66D1EB              <1>       shr    bx, 1        ; bx = month - 1 (0 to 11)
000065DB 6683FB01            <1>       cmp    bx, 1        ; if month > 2 and year mod 4  = 0
000065DF 76E6                <1>       jna    short cfe2   ; then mday = mday + 1
000065E1 08C0                <1>       or     al, al       ; if past 2/29 and leap year then
000065E3 75E2                <1>       jnz    short cfe2   ; add leap day (to mday)
000065E5 6642                <1>       inc    dx           ; mday = mday + 1
000065E7 EBDE                <1>       jmp    short cfe2
                             <1> cfe3:
000065E9 6643                <1>       inc    bx         ; -> bx = month, 1 to 12
000065EB 66891D[2C550100]    <1>       mov    [month], bx
000065F2 6629D1              <1>       sub    cx, dx     ; day = jday - mday + 1
000065F5 6641                <1>       inc    cx
000065F7 66890D[2E550100]    <1>       mov    [day], cx
                             <1>
                             <1>       ; eax, ebx, ecx, edx is changed at return
                             <1>       ; output ->
                             <1>       ; [year], [month], [day], [hour], [minute], [second]
                             <1>
000065FE C3                  <1>       retn   ; 31/12/2017 (TRDOS 386)
                             <1>
                             <1> set_rtc_date_time:
                             <1>       ; 31/12/2017
                             <1>       ; 30/12/2017 (TRDOS 386)
                             <1>       ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)
                             <1>       ; 20/06/2013 (Retro UNIX 8086 v1)
000065FF E80F000000          <1>       call   set_date_bcd
                             <1>       ; Set real-time clock date
00006604 E883F4FFFF          <1>       call   set_rtc_date ; RTC_50
                             <1>       ; Set real-time clock time
00006609 E832000000          <1>       call   set_time_bcd
0000660E E90AF4FFFF          <1>       jmp    set_rtc_time ; RTC_30
                             <1>
                             <1> ; 31/12/2017
                             <1> set_date_bcd:
00006613 A0[2B550100]        <1>         mov     al, [year+1]
00006618 D40A                <1>       aam    ; ah = al / 10, al = al mod 10
0000661A D510                <1>       db     0D5h,10h     ; Undocumented inst. AAD
                             <1>                         ; AL = AH * 10h + AL
0000661C 88C5                <1>       mov    ch, al ; century (BCD)
0000661E A0[2A550100]        <1>       mov    al, [year]
00006623 D40A                <1>       aam    ; ah = al / 10, al = al mod 10
00006625 D510                <1>       db     0D5h,10h     ; Undocumented inst. AAD
                             <1>                         ; AL = AH * 10h + AL
00006627 88C1                <1>       mov    cl, al ; year (BCD)
00006629 A0[2C550100]        <1>         mov al, [month]
0000662E D40A                <1>       aam    ; ah = al / 10, al = al mod 10
00006630 D510                <1>       db     0D5h,10h     ; Undocumented inst. AAD
                             <1>                         ; AL = AH * 10h + AL
```

```
00006632 88C6                <1>       mov    dh, al ; month (BCD)
00006634 A0[2E550100]        <1>       mov    al, [day]
00006639 D40A                <1>       aam    ; ah = al / 10, al = al mod 10
0000663B D510                <1>       db     0D5h,10h     ; Undocumented inst. AAD
                             <1>                          ; AL = AH * 10h + AL
0000663D 88C6                <1>       mov    dh, al ; day (BCD)
0000663F C3                  <1>       retn   ; 30/12/2017
                             <1> 
                             <1> ; 31/12/2017
                             <1> set_time_bcd:
                             <1>         ; Read real-time clock time
                             <1>       ; (get day light saving time bit status)
00006640 FA                  <1>       cli
00006641 E898F5FFFF          <1>       CALL   UPD_IPR             ; CHECK FOR UPDATE IN PROCESS
                             <1>       ; cf = 1 -> al = 0
00006646 7207                <1>         jc      short stime1
00006648 B00B                <1>       MOV    AL,CMOS_REG_B       ; ADDRESS ALARM REGISTER
0000664A E8AAF5FFFF          <1>       CALL   CMOS_READ           ; READ CURRENT VALUE OF DSE BIT
                             <1> stime1:
0000664F FB                  <1>       sti
00006650 2401                <1>       AND    AL,00000001B        ; MASK FOR VALID DSE BIT
00006652 88C2                <1>       MOV    DL,AL               ; SET [DL] TO ZERO FOR NO DSE BIT
                             <1>       ; DL = 1 or 0 (day light saving time)
                             <1>       ;
00006654 A0[30550100]        <1>       mov    al, [hour]
00006659 D40A                <1>       aam    ; ah = al / 10, al = al mod 10
0000665B D510                <1>       db     0D5h,10h     ; Undocumented inst. AAD
                             <1>                          ; AL = AH * 10h + AL
0000665D 88C5                <1>       mov    ch, al ; hour (BCD)
0000665F A0[32550100]        <1>         mov     al, [minute]
00006664 D40A                <1>       aam    ; ah = al / 10, al = al mod 10
00006666 D510                <1>       db     0D5h,10h     ; Undocumented inst. AAD
                             <1>                          ; AL = AH * 10h + AL
00006668 88C1                <1>       mov    cl, al       ; minute (BCD)
0000666A A0[34550100]        <1>         mov     al, [second]
0000666F D40A                <1>       aam    ; ah = al / 10, al = al mod 10
00006671 D510                <1>       db     0D5h,10h     ; Undocumented inst. AAD
                             <1>                          ; AL = AH * 10h + AL
00006673 88C6                <1>       mov    dh, al      ; second (BCD)
00006675 C3                  <1>       retn   ; 30/12/2017
                                 %include 'trdosk2.s' ; 04/01/2016
                             <1> ; *****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - DRV INIT : trdosk2.s
                             <1> ; -----------------------------------------------------------------------------
                             <1> ; Last Update: 27/12/2017
                             <1> ; -----------------------------------------------------------------------------
                             <1> ; Beginning: 04/01/2016
                             <1> ; -----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; -----------------------------------------------------------------------------
                             <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
                             <1> ; TRDOS2.ASM (09/11/2011)
                             <1> ; *****************************************************************************
                             <1> ; DRV_INIT.ASM (c) 2009-2011 Erdogan TAN  [26/09/2009] Last Update: 07/08/2011
                             <1> ;
                             <1> 
                             <1> ldrv_init: ; Logical Drive Initialization
                             <1>       ; 27/12/2017
                             <1>       ; 12/02/2016
                             <1>       ; 06/01/2016
                             <1>       ;      ('diskinit.inc', 'diskio.inc' integration)
                             <1>       ; 04/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 07/08/2011
                             <1>       ; 20/09/2009
                             <1>       ; 2005
00006676 0FB60D[D4580100]    <1>       movzx  ecx, byte [HF_NUM] ; number of fixed disks
0000667D 80F901              <1>       cmp    cl, 1
00006680 7301                <1>       jnb    short load_hd_partition_tables
                             <1>       ; No hard disks
00006682 C3                  <1>       retn
                             <1> load_hd_partition_tables:
00006683 8B35[D8580100]      <1>       mov    esi, [HDPM_TBL_VEC] ; primary master disk FDPT
00006689 BF[FE5C0100]        <1>       mov    edi, PTable_hd0
0000668E B280                <1>       mov    dl, 80h
                             <1> load_next_hd_partition_table:
00006690 51                  <1>       push   ecx
00006691 57                  <1>       push   edi
00006692 56                  <1>       push   esi ; FDPT (+ DPTE) address
00006693 8A4614              <1>       mov    al, [esi+20] ; DPTE offset 4
00006696 2440                <1>       and    al, 40h ;  LBA bit (bit 6)
                             <1>       ;shr   al, 6
00006698 A2[FF5E0100]        <1>       mov    [HD_LBA_yes], al
0000669D E82B040000          <1>       call   load_masterboot
000066A2 7275                <1>       jc     short pass_pt_this_hard_disk
                             <1> 
000066A4 BE[BC5C0100]        <1>       mov    esi, PartitionTable
000066A9 89F3                <1>       mov    ebx, esi
                             <1>       ;mov   ecx, 16
000066AB B110                <1>       mov    cl, 16
000066AD F3A5                <1>       rep    movsd
000066AF 89DE                <1>       mov    esi, ebx
000066B1 C605[F55C0000]04    <1>       mov    byte [hdc], 4 ; 4 - partition index
                             <1> loc_validate_hdp_partition:
000066B8 807E0400            <1>       cmp    byte [esi+ptFileSystemID], 0
000066BC 7641                <1>       jna    short loc_validate_next_hdp_partition2
000066BE 56                  <1>       push   esi ; Masterboot partition table offset
000066BF 52                  <1>       push   edx ; dl = Physical drive number
000066C0 FE05[005F0100]      <1>       inc    byte [PP_Counter]
000066C6 31FF                <1>         xor  edi, edi ; 0
                             <1>       ; Input -> ESI = PartitionTable offset
                             <1>       ; DL = Hard disk drive number
                             <1>       ; EDI = 0 -> Primary Partition
                             <1>       ; EDI > 0 -> Extended Partition's Start Sector
```

```
000066C8 E879010000          <1>       call   validate_hd_fat_partition
000066CD 730A                <1>       jnc    short loc_set_valid_hdp_partition_entry
                             <1>       ;pop   edx
                             <1>       ;push  edx
000066CF 8B1424              <1>       mov    edx, [esp]
000066D2 E8D4020000          <1>       call   validate_hd_fs_partition
000066D7 7224                <1>       jc     short loc_validate_next_hdp_partition1
                             <1> loc_set_valid_hdp_partition_entry:
000066D9 8A0D[D20C0100]      <1>       mov    cl, [Last_DOS_DiskNo]
000066DF 80C141              <1>       add    cl, 'A'
                             <1>       ; ESI = Logical dos drive description table address
000066E2 880E                <1>       mov    [esi+LD_Name], cl
000066E4 8A6602              <1>       mov    ah, [esi+LD_PhyDrvNo]
000066E7 88E0                <1>       mov    al, ah ; Physical drive number
000066E9 2C80                <1>       sub    al, 80h
000066EB C0E002              <1>       shl    al, 2
000066EE 0404                <1>       add    al, 4 ; 0 Based
000066F0 2A05[F55C0000]      <1>       sub    al, [hdc] ; 4 - partition index
                             <1>       ; AL = Partition entry/index, 0 based
                             <1>       ;  0 -> hd 0, Partition Table offset = 0
                             <1>       ; 15 -> hd 3, Partition Table offset = 3
                             <1>       ;mov   [esi+LD_PartitionEntry], al
000066F6 80EC7E              <1>       sub    ah, 7Eh
                             <1>       ; AH = Physical drive index, zero based
                             <1>       ;  0 for drive A:, 2 for drive C:
                             <1>       ;mov   [esi+LD_DParamEntry], ah
000066F9 6689467C            <1>       mov    [esi+LD_PartitionEntry], ax
                             <1> loc_validate_next_hdp_partition1:
000066FD 5A                  <1>       pop    edx ; dl = Physical drive number
000066FE 5E                  <1>       pop    esi ; Masterboot partition table offset
                             <1> loc_validate_next_hdp_partition2:
                             <1>       ; ESI = PartitionTable offset
                             <1>       ; DL = Hard/Fixed disk drive number
000066FF FE0D[F55C0000]      <1>       dec    byte [hdc] ; 4 - partition index
00006705 7412                <1>       jz     short pass_pt_this_hard_disk
00006707 83C610              <1>       add    esi, 16 ; 10h
0000670A EBAC                <1>       jmp    short loc_validate_hdp_partition
                             <1> loc_next_hd_partition_table:
0000670C FEC2                <1>       inc    dl
0000670E 83C620              <1>       add    esi, 32 ; next FDPT address
00006711 83C740              <1>       add    edi, 64 ; next partition table destination
00006714 E977FFFFFF          <1>         jmp     load_next_hd_partition_table
                             <1> pass_pt_this_hard_disk:
00006719 5E                  <1>       pop    esi ; FDPT (+ DPTE) address
0000671A 5F                  <1>       pop    edi ; Ptable_hd?
0000671B 59                  <1>       pop    ecx
0000671C E2EE                <1>       loop   loc_next_hd_partition_table
0000671E 803D[005F0100]01    <1>       cmp    byte [PP_Counter], 1
00006725 7301                <1>       jnb    short load_extended_dos_partitions
                             <1>       ; Empty partition table
00006727 C3                  <1>       retn
                             <1> load_extended_dos_partitions:
00006728 BE[FE5C0100]        <1>       mov    esi, PTable_hd0
0000672D BF[FE5D0100]        <1>       mov    edi, PTable_ep0
00006732 C605[F55C0000]80    <1>       mov    byte [hdc], 80h
                             <1> next_hd_extd_partition:
00006739 56                  <1>       push   esi ; PTable_hd? offset
0000673A 57                  <1>       push   edi ; PTable_ep?
                             <1>       ;mov   ecx, 4
0000673B B104                <1>       mov    cl, 4
0000673D 8A15[F55C0000]      <1>       mov    dl, byte [hdc]
                             <1> hd_check_fs_id_05h:
00006743 8A4604              <1>       mov    al, [esi+ptFileSystemID]
00006746 3C05                <1>       cmp    al, 05h ; Is it an extended dos partition ?
00006748 7404                <1>       je     short loc_set_ep_start_sector
0000674A 3C0F                <1>       cmp    al, 0Fh ; Is it an extended win4 (LBA mode) partition ?
0000674C 7546                <1>       jne    short continue_to_check_ep
                             <1> loc_set_ep_start_sector:
0000674E FE05[015F0100]      <1>       inc    byte [EP_Counter]
00006754 88D4                <1>       mov    ah, dl ; byte [hdc]
00006756 86E0                <1>       xchg   ah, al ; al = Drv Number, ah = Partition Identifier
00006758 50                  <1>       push   eax
00006759 30E4                <1>       xor    ah, ah
0000675B 2C80                <1>       sub    al, 80h
0000675D 50                  <1>       push   eax
0000675E C0E002              <1>       shl    al, 2 ; al = al * 4
00006761 0FB6D8              <1>       movzx  ebx, al
00006764 81C3[025F0100]      <1>       add    ebx, EP_StartSector
0000676A 8B4608              <1>       mov    eax, [esi+ptStartSector]
                             <1>         ; EAX = Extended partition's start sector
0000676D 8903                <1>         mov  [ebx], eax
0000676F 58                  <1>       pop    eax ; AL = Drv number - 80h, AH = 0
00006770 5A                  <1>       pop    edx ; DL = Drv number, DH = Partition ID
00006771 BB[FE5A0100]        <1>       mov    ebx, MasterBootBuff
00006776 803D[FF5E0100]01    <1>       cmp    byte [HD_LBA_yes], 1 ; LBA ready = Yes
0000677D 7240                <1>       jb     short loc_hd_load_ep_05h
0000677F 80FE05              <1>       cmp    dh, 05h
00006782 743B                <1>       je     short loc_hd_load_ep_05h
                             <1> loc_hd_load_ep_0Fh:
                             <1>       ; 04/01/2016
00006784 51                  <1>       push   ecx
00006785 8B4E08              <1>       mov    ecx, [esi+ptStartSector] ; sector number
                             <1>       ;mov   ebx, MasterBootBuff ; buffer address
                             <1>       ; LBA read/write (with private LBA function)
                             <1>       ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))
                             <1>       ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
00006788 B41B                <1>       mov    ah, 1Bh ; LBA read
0000678A B001                <1>       mov    al, 1 ; sector count
0000678C E875DAFFFF          <1>       call   int13h
00006791 59                  <1>       pop    ecx
00006792 733F                <1>       jnc    short loc_hd_move_ep_table
                             <1> continue_to_check_ep:
00006794 83C610              <1>       add    esi, 16
```

```
00006797 E2AA                <1>        loop   hd_check_fs_id_05h
                             <1> continue_check_ep_next_disk:
00006799 5F                  <1>        pop    edi ; PTable_ep?
0000679A 5E                  <1>        pop    esi ; PTable_hd?
0000679B A0[D4580100]        <1>        mov    al, [HF_NUM] ; number of hard disks
000067A0 047F                <1>        add    al, 7Fh
000067A2 3805[F55C0000]      <1>        cmp    [hdc], al
000067A8 0F8392000000        <1>        jnb    loc_validating_hd_partitions_ok
000067AE 83C640              <1>        add    esi, 64
000067B1 83C740              <1>        add    edi, 64
000067B4 FE05[F55C0000]      <1>        inc    byte [hdc]
000067BA E97AFFFFFF          <1>        jmp    next_hd_extd_partition
                             <1> loc_hd_load_ep_05h:
000067BF 51                  <1>        push   ecx
000067C0 8A7601              <1>        mov    dh, [esi+ptBeginHead]
000067C3 668B4E02            <1>          mov     cx, word [esi+ptBeginSector]
000067C7 66B80102            <1>        mov    ax, 0201h ; Read 1 sector
                             <1>        ;mov   ebx, MasterBootBuff
000067CB E836DAFFFF          <1>        call   int13h
000067D0 59                  <1>        pop    ecx
000067D1 72C1                <1>        jc     short continue_to_check_ep
                             <1> loc_hd_move_ep_table:
                             <1>          ;pop edi
                             <1>        ;push  edi  ; PTable_ep?
000067D3 8B3C24              <1>        mov    edi, [esp]
000067D6 BE[BC5C0100]        <1>          mov  esi, PartitionTable ; Extended
000067DB 89F3                <1>        mov    ebx, esi
                             <1>        ;mov   ecx, 16
000067DD B110                <1>        mov    cl, 16
000067DF F3A5                <1>               rep    movsd
000067E1 89DE                <1>        mov    esi, ebx
                             <1> loc_set_hde_sub_partition_count:
000067E3 C605[005F0100]04    <1>        mov    byte [PP_Counter], 4
                             <1> loc_validate_hde_partition:
000067EA 807E0400            <1>        cmp    byte [esi+ptFileSystemID], 0
000067EE 763F                <1>        jna    short loc_validate_next_hde_partition2
000067F0 56                  <1>        push   esi ; Extended partition table offset
000067F1 8A15[F55C0000]      <1>        mov    dl, byte [hdc]
000067F7 0FB6C2              <1>        movzx  eax, dl
000067FA 2C80                <1>        sub    al, 80h
000067FC C0E002              <1>        shl    al, 2
                             <1>        ; 06/01/2016
                             <1>        ; (TRDOS v1.0 had a bug here, in 'DRV_INIT.ASM')
                             <1>        ; BUGFIX *
                             <1>        ;mov   ecx, eax
000067FF 88C1                <1>        mov    cl, al
00006801 80C104              <1>        add    cl, 4
00006804 2A0D[005F0100]      <1>        sub    cl, [PP_Counter] ; 4 to 1
                             <1>        ; CL = Partition entry/index, 0 based
                             <1>          ;  0 -> hd 0, Partition Table offset = 0
                             <1>          ; 15 -> hd 3, Partition Table offset = 3
0000680A 88D5                <1>               mov    ch, dl
0000680C 80ED7E              <1>        sub    ch, 7Eh ;
                             <1>        ; CH = Physical drive index, zero based
                             <1>        ;  0 for drive A:, 2 for drive C:
                             <1>        ; BUGFIX *
0000680F 51                  <1>        push   ecx ; *
00006810 BF[025F0100]        <1>        mov    edi, EP_StartSector
00006815 01C7                <1>        add    edi, eax
                             <1>        ; Input -> ESI = PartitionTable offset
                             <1>        ; DL = Hard disk drive number
                             <1>        ; EDI = Extended partition start sector pointer
00006817 E82A000000          <1>        call   validate_hd_fat_partition
0000681C 59                  <1>        pop    ecx ; *
0000681D 720F                <1>        jc     short loc_validate_next_hde_partition1
                             <1> loc_set_valid_hde_partition_entry:
                             <1>        ; 06/01/2016 (TRDOS v2.0)
                             <1>        ; BUGFIX *
                             <1>        ;mov   [esi+LD_PartitionEntry], cl
                             <1>        ;mov   [esi+LD_DParamEntry], ch
0000681F 66894E7C            <1>        mov    [esi+LD_PartitionEntry], cx
                             <1>        ;
00006823 8A0D[D20C0100]      <1>        mov    cl, [Last_DOS_DiskNo]
00006829 80C141              <1>        add    cl, 'A'
0000682C 880E                <1>        mov    [esi+LD_Name], cl
                             <1> loc_validate_next_hde_partition1:
0000682E 5E                  <1>        pop    esi ; Extended partition table offset
                             <1> loc_validate_next_hde_partition2:
                             <1>        ; ESI = Extended partition table offset
                             <1>        ; DL = Hard disk drive number
0000682F FE0D[005F0100]      <1>        dec    byte [PP_Counter]
00006835 0F845EFFFFFF        <1>        jz     continue_check_ep_next_disk
0000683B 83C610              <1>        add    esi, 16 ; 10h
0000683E EBAA                <1>        jmp    short loc_validate_hde_partition
                             <1> loc_validating_hd_partitions_ok:
00006840 A0[D20C0100]        <1>        mov    al, [Last_DOS_DiskNo]
                             <1> loc_drv_init_retn:
00006845 C3                  <1>        retn
                             <1>
                             <1> validate_hd_fat_partition:
                             <1>        ; 27/12/2017
                             <1>        ; 12/02/2016
                             <1>        ; 07/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>        ; 07/08/2011
                             <1>        ; 23/07/2011
                             <1>        ; Input
                             <1>        ;   DL = Hard/Fixed Disk Drive Number
                             <1>        ;   ESI = PartitionTable offset
                             <1>        ;   EDI = Extend. Part. Start Sector Pointer
                             <1>        ;   EDI = 0 -> Primary Partition
                             <1>        ;   byte [Last_DOS_DiskNo]
                             <1>        ; Output
                             <1>        ;  cf=0 -> Validated
                             <1>       ;   ESI = Logical dos drv desc. table
                             <1>       ;   EBX = FAT boot sector buffer
                             <1>       ;   byte [Last_DOS_DiskNo]
                             <1>       ;  cf=1 -> Not a valid FAT partition
                             <1>       ; EAX, EDX, ECX, EDI -> changed
                             <1>
                             <1>       ;mov   esi, PartitionTable
00006846 8A6604              <1>       mov    ah, [esi+ptFileSystemID]
00006849 B002                <1>       mov    al, 2 ; 27/12/2017
0000684B 80FC06              <1>       cmp    ah, 06h ; FAT16 CHS partition
                             <1>       ; 12/02/2016
                             <1>       ;jb    short loc_not_a_valid_fat_partition2
0000684E 7310                <1>       jnb    short vhdp_FAT16_32
                             <1>       ;
                             <1>       ; 27/12/2017
00006850 FEC8                <1>       dec    al ; mov al, 1
00006852 38C4                <1>       cmp    ah, al ; 1 ; FAT12 partition
00006854 7421                <1>       je     short loc_set_valid_hd_partition_params
                             <1>       ;
00006856 FEC0                <1>       inc    al ; mov al, 2
00006858 80FC04              <1>       cmp    ah, 04h ; FAT16 CHS partition (< 32MB)
0000685B 741A                <1>       je     short loc_set_valid_hd_partition_params
0000685D 7716                <1>       ja     short loc_not_a_valid_fat_partition1
                             <1>       ; cf=1
0000685F C3                  <1>       retn
                             <1> vhdp_FAT16_32:
00006860 80FC0E              <1>       cmp    ah, 0Eh ; FAT16 LBA partition
00006863 7710                <1>       ja     short loc_not_a_valid_fat_partition1
00006865 7410                <1>       je     short loc_set_valid_hd_partition_params
                             <1>       ;mov   al, 3
00006867 FEC0                <1>       inc    al
00006869 80FC0B              <1>       cmp    ah, 0Bh ; FAT32 CHS partition
0000686C 7409                <1>       je     short loc_set_valid_hd_partition_params
0000686E 7206                <1>       jb     short loc_not_a_valid_fat_partition2
00006870 80FC0C              <1>       cmp    ah, 0Ch ; FAT32 LBA partition
00006873 7402                <1>       je     short loc_set_valid_hd_partition_params
                             <1> loc_not_a_valid_fat_partition1:
00006875 F9                  <1>       stc
                             <1> loc_not_a_valid_fat_partition2:
00006876 C3                  <1>       retn
                             <1>
                             <1> loc_set_valid_hd_partition_params:
00006877 FE05[D20C0100]      <1>       inc    byte [Last_DOS_DiskNo] ; > 1
                             <1>       ;
0000687D 31DB                <1>       xor    ebx, ebx
0000687F 8A3D[D20C0100]      <1>       mov    bh, [Last_DOS_DiskNo] ; * 256
00006885 81C300010900        <1>       add    ebx, Logical_DOSDisks
                             <1>       ;
0000688B C6430102            <1>       mov    byte [ebx+LD_DiskType], 2
0000688F 885302              <1>       mov    byte [ebx+LD_PhyDrvNo], dl
                             <1>       ;mov   byte [ebx+LD_FATType], al ; 2 or 3
                             <1>       ;mov   byte [ebx+LD_FSType], ah ; 06h, 0Eh, 0Bh, 0Ch
00006892 66894303            <1>       mov    word [ebx+LD_FATType], ax
                             <1>       ;
00006896 8B4E08              <1>       mov    ecx, [esi+ptStartSector]
00006899 09FF                <1>       or     edi, edi
0000689B 7402                <1>       jz     short pass_hd_FAT_ep_start_sector_adding
                             <1> loc_add_hd_FAT_ep_start_sector:
0000689D 030F                <1>       add    ecx, [edi]
                             <1> pass_hd_FAT_ep_start_sector_adding:
0000689F 894B6C              <1>       mov    [ebx+LD_StartSector], ecx
                             <1> loc_hd_FAT_logical_drv_init:
000068A2 89DD                <1>       mov    ebp, ebx
                             <1>       ;mov   dl, [ebx+LD_PhyDrvNo]
000068A4 A0[FF5E0100]        <1>       mov    al, [HD_LBA_yes] ; 07/01/2016
000068A9 884305              <1>       mov    [ebx+LD_LBAYes], al
000068AC BB[125F0100]        <1>       mov    ebx, DOSBootSectorBuff ; buffer address
000068B1 08C0                <1>       or     al, al
000068B3 740C                <1>       jz     short loc_hd_FAT_drv_init_load_bs_chs
                             <1> loc_hd_FAT_drv_init_load_bs_lba:
                             <1>       ; DL = Physical drive number
                             <1>       ;mov   ecx, [esi+ptStartSector] ; sector number
                             <1>       ;mov   ebx, DOSBootSectorBuff ; buffer address
                             <1>       ; LBA read/write (with private LBA function)
                             <1>       ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))
                             <1>       ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
000068B5 B41B                <1>       mov    ah, 1Bh ; LBA read
000068B7 B001                <1>       mov    al, 1 ; sector count
000068B9 E848D9FFFF          <1>       call   int13h
000068BE 7313                <1>       jnc    short loc_hd_drv_FAT_boot_validation
                             <1> loc_not_a_valid_fat_partition3:
000068C0 C3                  <1>       retn
                             <1> loc_hd_FAT_drv_init_load_bs_chs:
000068C1 8A7601              <1>       mov    dh, [esi+ptBeginHead]
000068C4 668B4E02            <1>       mov    cx, [esi+ptBeginSector]
000068C8 66B80102            <1>       mov    ax, 0201h ; Read 1 sector
                             <1>       ;mov   ebx, DOSBootSectorBuff
000068CC E835D9FFFF          <1>       call   int13h
000068D1 72ED                <1>       jc     short loc_not_a_valid_fat_partition3
                             <1> loc_hd_drv_FAT_boot_validation:
                             <1>       ;mov   esi, DOSBootSectorBuff
000068D3 89DE                <1>       mov    esi, ebx
000068D5 6681BEFE01000055AA  <1>       cmp    word [esi+BS_Validation], 0AA55h
000068DE 7512                <1>       jne    short loc_not_a_valid_fat_partition4
000068E0 807E15F8            <1>       cmp    byte [esi+BPB_Media], 0F8h
000068E4 750C                <1>       jne    short loc_not_a_valid_fat_partition4
                             <1>
                             <1>       ; 27/12/2017
000068E6 807D0303            <1>       cmp    byte [ebp+LD_FATType], 3
000068EA 7508                <1>       jne    short loc_hd_FAT16_BPB
                             <1>
                             <1> loc_hd_drv_FAT32_boot_validation:
000068EC 807E4229            <1>       cmp    byte [esi+BS_FAT32_BootSig], 29h
```

```
000068F0 7416                <1>       je     short loc_hd_FAT32_BPB
                             <1> 
                             <1> loc_not_a_valid_fat_partition4:
000068F2 F9                  <1>       stc
000068F3 C3                  <1>       retn
                             <1> 
                             <1> loc_hd_FAT16_BPB:
000068F4 807E2629            <1>       cmp    byte [esi+BS_BootSig], 29h
000068F8 75F8                <1>       jne    short loc_not_a_valid_fat_partition4
                             <1> 
000068FA 66837E1600          <1>       cmp    word [esi+BPB_FATSz16], 0
000068FF 7607                <1>       jna    short loc_hd_big_FAT16_BPB
00006901 B920000000          <1>       mov    ecx, 32
00006906 EB05                <1>       jmp    short loc_hd_move_FAT_BPB
                             <1> 
                             <1> loc_hd_FAT32_BPB:
                             <1>       ;cmp   word [esi+BPB_FATSz16], 0
                             <1>       ;ja    short loc_not_a_valid_fat_partition4
                             <1> loc_hd_big_FAT16_BPB:
00006908 B92D000000          <1>       mov    ecx, 45
                             <1> loc_hd_move_FAT_BPB:
0000690D 89EF                <1>       mov    edi, ebp
                             <1>       ;mov   esi, ebx ; Boot sector
0000690F 57                  <1>       push   edi
00006910 83C706              <1>       add    edi, LD_BPB
00006913 F366A5              <1>       rep    movsw
00006916 5E                  <1>       pop    esi
00006917 0FB74614            <1>       movzx  eax, word [esi+LD_BPB+BPB_RsvdSecCnt]
0000691B 03466C              <1>       add    eax, [esi+LD_StartSector]
0000691E 894660              <1>       mov    [esi+LD_FATBegin], eax
00006921 807E0303            <1>       cmp    byte [esi+LD_FATType], 3
00006925 7224                <1>       jb     short loc_set_FAT16_RootDirLoc
                             <1> loc_set_FAT32_RootDirLoc:
00006927 8B462A              <1>       mov    eax, [esi+LD_BPB+BPB_FATSz32]
0000692A 0FB65E16            <1>         movzx      ebx, byte [esi+LD_BPB+BPB_NumFATs]
0000692E F7E3                <1>       mul    ebx
00006930 034660              <1>       add    eax, [esi+LD_FATBegin]
                             <1> loc_set_FAT32_data_begin:
00006933 894668              <1>       mov    [esi+LD_DATABegin], eax
00006936 894664              <1>       mov    [esi+LD_ROOTBegin], eax
                             <1>       ; If Root Directory Cluster <> 2 then
                             <1>       ; change the beginning sector value
                             <1>       ; of the root dir by adding sector offset.
00006939 8B4632              <1>       mov    eax, [esi+LD_BPB+BPB_RootClus]
0000693C 83E802              <1>       sub    eax, 2
0000693F 7442                <1>       jz     short short loc_set_32bit_FAT_total_sectors
                             <1>       ;movzx ebx, byte [esi+LD_BPB+BPB_SecPerClust]
00006941 8A5E13              <1>       mov    bl, byte [esi+LD_BPB+BPB_SecPerClust]
00006944 F7E3                <1>       mul    ebx
00006946 014664              <1>       add    [esi+LD_ROOTBegin], eax
00006949 EB38                <1>       jmp    short loc_set_32bit_FAT_total_sectors
                             <1>       ;
                             <1> loc_set_FAT16_RootDirLoc:
0000694B 0FB64616            <1>       movzx  eax, byte [esi+LD_BPB+BPB_NumFATs]
0000694F 0FB7561C            <1>       movzx  edx, word [esi+LD_BPB+BPB_FATSz16]
00006953 F7E2                <1>       mul    edx
00006955 034660              <1>       add    eax, [esi+LD_FATBegin]
00006958 894664              <1>       mov    [esi+LD_ROOTBegin], eax
                             <1> loc_set_FAT16_data_begin:
0000695B 894668              <1>       mov    [esi+LD_DATABegin], eax
0000695E B820000000          <1>       mov    eax, 20h  ; Size of a directory entry
                             <1>       ;movzx edx, word [esi+LD_BPB+BPB_RootEntCnt]
00006963 668B5617            <1>         mov     dx, [esi+LD_BPB+BPB_RootEntCnt]
00006967 F7E2                <1>         mul  edx
                             <1>       ;mov   ecx, 511
00006969 66B9FF01            <1>       mov    cx, 511
0000696D 01C8                <1>       add    eax, ecx
0000696F 41                  <1>       inc    ecx ; 512
00006970 F7F1                <1>       div    ecx
00006972 014668              <1>       add    [esi+LD_DATABegin], eax
00006975 0FB74619            <1>       movzx  eax, word [esi+LD_BPB+BPB_TotalSec16]
00006979 6685C0              <1>       test   ax, ax
0000697C 7405                <1>       jz     short loc_set_32bit_FAT_total_sectors
                             <1> loc_set_16bit_FAT_total_sectors:
0000697E 894670              <1>       mov    [esi+LD_TotalSectors], eax
00006981 EB06                <1>       jmp    short loc_set_hd_FAT_cluster_count
                             <1> loc_set_32bit_FAT_total_sectors:
00006983 8B4626              <1>       mov    eax, [esi+LD_BPB+BPB_TotalSec32]
00006986 894670              <1>       mov    [esi+LD_TotalSectors], eax
                             <1> loc_set_hd_FAT_cluster_count:
00006989 8B5668              <1>       mov    edx, [esi+LD_DATABegin]
0000698C 2B566C              <1>       sub    edx, [esi+LD_StartSector]
0000698F 29D0                <1>       sub    eax, edx
00006991 31D2                <1>       xor    edx, edx ; 0
00006993 0FB64E13            <1>         movzx   ecx, byte [esi+LD_BPB+BPB_SecPerClust]
00006997 F7F1                <1>         div  ecx
00006999 894678              <1>       mov    [esi+LD_Clusters], eax
                             <1>       ; Maximum Valid Cluster Number= EAX +1
                             <1>       ; with 2 reserved clusters= EAX +2
                             <1> loc_set_hd_FAT_fs_free_sectors:
                             <1>       ;mov   dword [esi+LD_FreeSectors], 0
0000699C E859010000          <1>       call   get_free_FAT_sectors
000069A1 7207                <1>       jc     short loc_validate_hd_FAT_partition_retn
000069A3 894674              <1>       mov    [esi+LD_FreeSectors], eax
000069A6 C6467E06            <1>       mov    byte [esi+LD_MediaChanged], 6  ; Volume Name Reset
                             <1>       ;mov   cl, [Last_DOS_DiskNo]
                             <1>       ;add   cl, 'A'
                             <1>       ;mov   [esi+LD_FS_Name], cl
                             <1> 
                             <1> loc_validate_hd_FAT_partition_retn:
000069AA C3                  <1>       retn
                             <1> 
                             <1> validate_hd_fs_partition:
```

```
                             <1>       ; 09/12/2017
                             <1>       ; 13/02/2016
                             <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 29/01/2011
                             <1>       ; 23/07/2011
                             <1>       ; Input
                             <1>       ;   DL = Hard/Fixed Disk Drive Number
                             <1>       ;   ESI = PartitionTable offset
                             <1>       ;   byte [Last_DOS_DiskNo]
                             <1>       ; Output
                             <1>       ;  cf=0 -> Validated
                             <1>       ;   ESI = Logical dos drv desc. table
                             <1>       ;   EBX = Singlix FS boot sector buffer
                             <1>       ;   byte [Last_DOS_DiskNo]
                             <1>       ;  cf=1 -> Not a valid 'Singlix FS' partition
                             <1>       ; EAX, EDX, ECX, EDI -> changed
                             <1>
                             <1>       ;mov   esi, PartitionTable
000069AB 8A6604              <1>       mov    ah, [esi+ptFileSystemID]
000069AE 80FCA1              <1>       cmp    ah, 0A1h ; SINGLIX FS1 (trfs1) partition
000069B1 7549                <1>       jne    short loc_validate_hd_fs_partition_stc_retn
                             <1> loc_set_valid_hd_fs_partition_params:
000069B3 FE05[D20C0100]      <1>       inc    byte [Last_DOS_DiskNo] ; > 1
000069B9 30C0                <1>       xor    al, al ; mov al, 0
                             <1>       ;mov   [drv], dl
000069BB 29DB                <1>       sub    ebx, ebx ; 0
000069BD 8A3D[D20C0100]      <1>       mov    bh, [Last_DOS_DiskNo]
000069C3 81C300010900        <1>       add    ebx, Logical_DOSDisks
000069C9 C6430102            <1>       mov    byte [ebx+LD_DiskType], 2
000069CD 885302              <1>       mov    [ebx+LD_PhyDrvNo], dl
                             <1>       ;mov   [ebx+LD_FATType], al ; 0
                             <1>       ;mov   [ebx+LD_FSType], ah
000069D0 66894303            <1>       mov    [ebx+LD_FATType], ax
                             <1>       ;mov   eax, [esi+ptStartSector]
                             <1>       ;mov   [ebx+LD_StartSector], eax
                             <1> loc_hd_fs_logical_drv_init:
000069D4 89DD                <1>       mov    ebp, ebx ; 10/01/2016
                             <1>       ;mov   dl, [ebx+LD_PhyDrvNo]
000069D6 A0[FF5E0100]        <1>       mov    al, [HD_LBA_yes] ; 10/01/2016
000069DB 884305              <1>       mov    [ebx+LD_LBAYes], al
000069DE 89DE                <1>       mov    esi, ebx
000069E0 BB[125F0100]        <1>       mov    ebx, DOSBootSectorBuff ; buffer address
000069E5 08C0                <1>       or     al, al
000069E7 7515                <1>       jnz    short loc_hd_fs_drv_init_load_bs_lba
                             <1> loc_hd_fs_drv_init_load_bs_chs:
000069E9 8A7601              <1>       mov    dh, [esi+ptBeginHead]
000069EC 668B4E02            <1>       mov    cx, [esi+ptBeginSector]
000069F0 66B80102            <1>       mov    ax, 0201h ; Read 1 sector
                             <1>       ;mov   ebx, DOSBootSectorBuff
000069F4 E80DD8FFFF          <1>       call   int13h
000069F9 7311                <1>       jnc    short loc_hd_drv_fs_boot_validation
                             <1> loc_validate_hd_fs_partition_err_retn:
000069FB C3                  <1>       retn
                             <1> loc_validate_hd_fs_partition_stc_retn:
000069FC F9                  <1>       stc
000069FD C3                  <1>       retn
                             <1> loc_hd_fs_drv_init_load_bs_lba:
                             <1>       ; DL = Physical drive number
                             <1>       ;mov   esi, ebx
000069FE 8B4E08              <1>       mov    ecx, [esi+ptStartSector] ; sector number
                             <1>       ;mov   ebx, DOSBootSectorBuff ; buffer address
                             <1>       ; LBA read/write (with private LBA function)
                             <1>       ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))
                             <1>       ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
00006A01 B41B                <1>       mov    ah, 1Bh ; LBA read
00006A03 B001                <1>       mov    al, 1 ; sector count
00006A05 E8FCD7FFFF          <1>       call   int13h
00006A0A 72EF                <1>       jc     short loc_validate_hd_fs_partition_err_retn
                             <1> loc_hd_drv_fs_boot_validation:
                             <1>       ;mov   esi, DOSBootSectorBuff
00006A0C 89DE                <1>       mov    esi, ebx ; Boot sector buffer
00006A0E 6681BEFE01000055AA  <1>       cmp    word [esi+BS_Validation], 0AA55h
00006A17 75E3                <1>       jne    short loc_validate_hd_fs_partition_stc_retn
                             <1>         ;
                             <1>       ;Singlix FS Extensions to TR-DOS (7/6/2009)
00006A19 66817E035346        <1>       cmp    word [esi+bs_FS_Identifier], 'SF'
00006A1F 75DB                <1>       jne    short loc_validate_hd_fs_partition_stc_retn
                             <1>         ;'A1h' check is not necessary
                             <1>       ;  if 'FS' check is passed as OK/Yes.
00006A21 807E09A1            <1>       cmp    byte [esi+bs_FS_PartitionID], 0A1h
00006A25 75D5                <1>       jne    short loc_validate_hd_fs_partition_stc_retn
                             <1>       ;
00006A27 89EF                <1>       mov    edi, ebp ; 10/01/2016
                             <1>       ;
00006A29 8A462D              <1>       mov    al, byte [esi+bs_FS_LBA_Ready]
00006A2C 884705              <1>       mov    [edi+LD_FS_LBAYes], al
                             <1>       ;
                             <1>       ; 03/01/2010 CHS -> DOS FAT/BPB compatibility fix
00006A2F 8A4608              <1>       mov    al, [esi+bs_FS_MediaAttrib]
00006A32 884706              <1>       mov    byte [edi+LD_FS_MediaAttrib], al
                             <1>       ;
00006A35 8A460A              <1>       mov    al, [esi+bs_FS_VersionMaj]
00006A38 884707              <1>       mov    [edi+LD_FS_VersionMajor], al
                             <1>       ;
00006A3B 668B4606            <1>       mov    ax, [esi+bs_FS_BytesPerSec]
00006A3F 66894711            <1>       mov    [edi+LD_FS_BytesPerSec], ax
00006A43 8A462E              <1>       mov    al, [esi+bs_FS_SecPerTrack]
00006A46 30E4                <1>       xor    ah, ah ; 09/12/2017
00006A48 6689471E            <1>       mov    [edi+LD_FS_SecPerTrack], ax
00006A4C 8A462F              <1>       mov    al, [esi+bs_FS_Heads]
00006A4F 66894720            <1>       mov    [edi+LD_FS_NumHeads], ax
                             <1>       ;
00006A53 8B4628              <1>       mov    eax, [esi+bs_FS_UnDelDirD]
```

```
00006A56 894722              <1>       mov    [edi+LD_FS_UnDelDirD], eax
00006A59 8B5618              <1>       mov    edx, [esi+bs_FS_MATLocation]
00006A5C 89570C              <1>       mov    [edi+LD_FS_MATLocation], edx
00006A5F 8B461C              <1>       mov    eax, [esi+bs_FS_RootDirD]
00006A62 894708              <1>       mov    [edi+LD_FS_RootDirD], eax
00006A65 8B460C              <1>       mov    eax, [esi+bs_FS_BeginSector]
00006A68 89476C              <1>       mov    [edi+LD_FS_BeginSector], eax
00006A6B 8B4710              <1>       mov    eax, [edi+bs_FS_VolumeSize]
00006A6E 894770              <1>       mov    [edi+LD_FS_VolumeSize], eax
                             <1>       ;
00006A71 89D0                <1>       mov    eax, edx ; [edi+LD_FS_MATLocation]
00006A73 03476C              <1>       add    eax, [edi+LD_FS_BeginSector]
00006A76 89FE                <1>       mov    esi, edi
                             <1> mread_hd_fs_MAT_sector:
                             <1>        ;mov  ebx, DOSBootSectorBuff
00006A78 B901000000          <1>       mov    ecx, 1
00006A7D E8568D0000          <1>       call   disk_read
00006A82 7248                <1>       jc     short loc_validate_hd_fs_partition_retn
                             <1>       ; EDI will not be changed
00006A84 89DE                <1>       mov    esi, ebx
                             <1> use_hdfs_mat_sector_params:
00006A86 8B460C              <1>       mov    eax, [esi+FS_MAT_DATLocation]
00006A89 894714              <1>       mov    [edi+LD_FS_DATLocation], eax
00006A8C 8B4610              <1>       mov    eax, [esi+FS_MAT_DATScount]
00006A8F 894718              <1>       mov    [edi+LD_FS_DATSectors], eax
00006A92 8B4614              <1>       mov    eax, [esi+FS_MAT_FreeSectors]
00006A95 894774              <1>         mov     [edi+LD_FS_FreeSectors], eax
00006A98 8B4618              <1>       mov    eax, [esi+FS_MAT_FirstFreeSector]
00006A9B 894778              <1>       mov    [edi+LD_FS_FirstFreeSector], eax
00006A9E 8B4708              <1>       mov    eax, [edi+LD_FS_RootDirD]
00006AA1 03476C              <1>       add    eax, [edi+LD_FS_BeginSector]
00006AA4 89FE                <1>       mov    esi, edi
                             <1> read_hd_fs_RDT_sector:
00006AA6 BB[125F0100]        <1>       mov    ebx, DOSBootSectorBuff
                             <1>       ;mov   ecx, 1
00006AAB B101                <1>       mov    cl, 1
00006AAD E8268D0000          <1>       call   disk_read
00006AB2 7218                <1>       jc     short loc_validate_hd_fs_partition_retn
                             <1>       ; EDI will not be changed
00006AB4 89DE                <1>       mov    esi, ebx
                             <1> use_hdfs_RDT_sector_params:
00006AB6 8B461C              <1>       mov    eax, [esi+FS_RDT_VolumeSerialNo]
00006AB9 894728              <1>       mov    [edi+LD_FS_VolumeSerial], eax
00006ABC 57                  <1>       push   edi
                             <1>       ;mov   ecx, 16
00006ABD B110                <1>       mov    cl, 16
00006ABF 83C640              <1>       add    esi, FS_RDT_VolumeName
00006AC2 83C72C              <1>       add    edi, LD_FS_VolumeName
00006AC5 F3A5                <1>       rep    movsd ; 64 bytes
00006AC7 5E                  <1>       pop    esi
                             <1>              ; Volume Name Reset
00006AC8 C6467E06            <1>         mov     byte [esi+LD_FS_MediaChanged], 6
                             <1>       ;
                             <1>         ;mov cl, [Last_DOS_DiskNo]
                             <1>       ;add   cl, 'A'
                             <1>       ;mov   [esi+LD_FS_Name], cl
                             <1>
                             <1> loc_validate_hd_fs_partition_retn:
00006ACC C3                  <1>       retn
                             <1>
                             <1> load_masterboot:
                             <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 2005 - 2011
                             <1>       ; input -> DL = drive number
00006ACD B40D                <1>       mov    ah, 0Dh ; Alternate disk reset
00006ACF E832D7FFFF          <1>       call   int13h
00006AD4 7301                <1>       jnc    short pass_reset_error
                             <1> harddisk_error:
00006AD6 C3                  <1>       retn
                             <1> pass_reset_error:
00006AD7 BB[FE5A0100]        <1>       mov    ebx, MasterBootBuff
00006ADC 66B80102            <1>       mov    ax, 0201h
00006AE0 66B90100            <1>       mov    cx, 1
00006AE4 30F6                <1>       xor    dh, dh
00006AE6 E81BD7FFFF          <1>       call   int13h
00006AEB 72E9                <1>       jc     short harddisk_error
                             <1>       ;
00006AED 66813D[FC5C0100]55- <1>       cmp    word [MBIDCode], 0AA55h
00006AF5 AA                  <1>
00006AF6 7401                <1>       je     short load_masterboot_ok
00006AF8 F9                  <1>       stc
                             <1> load_masterboot_ok:
00006AF9 C3                  <1>       retn
                             <1>
                             <1> get_free_FAT_sectors:
                             <1>       ; 21/12/2017
                             <1>       ; 29/02/2016
                             <1>       ; 13/02/2016
                             <1>       ; 04/02/2016
                             <1>       ; 07/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 11/07/2010
                             <1>       ; 21/06/2009
                             <1>       ; INPUT: ESI = Logical DOS Drive Description Table address
                             <1>       ; OUTPUT: STC => Error
                             <1>         ;    cf = 0 and EAX = Free FAT sectors
                             <1>       ; Also, related parameters and FAT buffer will be reset and updated
                             <1>
00006AFA 31C0                <1>       xor    eax, eax
                             <1>       ;mov   [esi+LD_FreeSectors], eax ; Reset
                             <1>
00006AFC 807E0302            <1>         cmp     byte [esi+LD_FATType], 2
00006B00 7654                <1>       jna    short loc_gfc_get_fat_free_clusters
                             <1>
                             <1>       ; 29/02/2016
00006B02 48                  <1>       dec    eax ; 0FFFFFFFFh
00006B03 89463A              <1>       mov    [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count (reset)
00006B06 89463E              <1>       mov    [esi+LD_BPB+BPB_Reserved+4], eax ; First Free Cluster (reset)
00006B09 40                  <1>       inc    eax ; 0
                             <1>       ;
00006B0A 668B4636            <1>       mov    ax, [esi+LD_BPB+BPB_FSInfo]
00006B0E 03466C              <1>       add    eax, [esi+LD_StartSector]
                             <1>
00006B11 BB[125F0100]        <1>       mov    ebx, DOSBootSectorBuff
00006B16 B901000000          <1>       mov    ecx, 1
00006B1B E8B88C0000          <1>       call   disk_read
00006B20 7301                <1>       jnc    short loc_gfc_check_fsinfo_signs
                             <1> retn_gfc_get_fsinfo_sec:
00006B22 C3                  <1>       retn
                             <1>
                             <1> loc_gfc_check_fsinfo_signs:
00006B23 BB[125F0100]        <1>       mov    ebx, DOSBootSectorBuff ; 13/02/2016
00006B28 813B52526141        <1>         cmp     dword [ebx], 41615252h
00006B2E 7524                <1>       jne    short retn_gfc_get_fsinfo_stc
                             <1>       ;add   ebx, 484
                             <1>       ;cmp   dword [ebx], 61417272h
00006B30 81BBE4010000727241- <1>       cmp    dword [ebx+484], 61417272h
00006B39 61                  <1>
00006B3A 7518                <1>       jne    short retn_gfc_get_fsinfo_stc
                             <1>       ;add   ebx, 4
                             <1>       ;mov   eax, [ebx]
00006B3C 8B83E8010000        <1>       mov    eax, [ebx+488]
                             <1>       ; 29/02/2016
00006B42 89463A              <1>       mov    [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count
00006B45 8B93EC010000        <1>       mov    edx,  [ebx+492]
00006B4B 89463E              <1>       mov    [esi+LD_BPB+BPB_Reserved+4], eax ; First Free Cluster
                             <1>       ; 21/12/2017
00006B4E 89C3                <1>       mov    ebx, eax ; (initial value = 0FFFFFFFFh)
00006B50 43                  <1>       inc    ebx ; 0FFFFFFFFh -> 0
00006B51 7513                <1>       jnz    short short retn_from_get_free_fat32_clusters
00006B53 C3                  <1>       retn
                             <1>
                             <1> retn_gfc_get_fsinfo_stc:
00006B54 F9                  <1>       stc
00006B55 C3                  <1>       retn
                             <1>
                             <1> loc_gfc_get_fat_free_clusters:
                             <1>       ;mov   eax, 2
00006B56 B002                <1>       mov    al, 2
                             <1>       ;mov   [FAT_CurrentCluster], eax
                             <1> loc_gfc_loop_get_next_cluster:
00006B58 E8EB4F0000          <1>       call   get_next_cluster
00006B5D 730E                <1>       jnc    short loc_gfc_free_fat_clusters_cont
00006B5F 21C0                <1>       and    eax, eax
00006B61 7411                <1>       jz     short loc_gfc_pass_inc_free_cluster_count
                             <1>
                             <1> retn_from_get_free_fat_clusters:
00006B63 8B4674              <1>       mov    eax, [esi+LD_FreeSectors] ; Free clusters !
                             <1> retn_from_get_free_fat32_clusters:
00006B66 0FB65E13            <1>         movzx      ebx, byte [esi+LD_BPB+BPB_SecPerClust]
00006B6A F7E3                <1>             mul    ebx
                             <1>       ;mov   [esi+LD_FreeSectors], eax ; Free sectors
                             <1> retn_get_free_sectors_calc:
00006B6C C3                  <1>       retn
                             <1>
                             <1> loc_gfc_free_fat_clusters_cont:
00006B6D 09C0                <1>       or     eax, eax
00006B6F 7503                <1>       jnz    short loc_gfc_pass_inc_free_cluster_count
00006B71 FF4674              <1>       inc    dword [esi+LD_FreeSectors] ; Free clusters !
                             <1>
                             <1> loc_gfc_pass_inc_free_cluster_count:
                             <1>       ;mov   eax, [FAT_CurrentCluster]
00006B74 89C8                <1>       mov    eax, ecx ; [FAT_CurrentCluster]
00006B76 3B4678              <1>       cmp    eax, [esi+LD_Clusters]
00006B79 77E8                <1>       ja     short retn_from_get_free_fat_clusters
00006B7B 40                  <1>       inc    eax
                             <1>       ;mov   [FAT_CurrentCluster], eax
00006B7C EBDA                <1>       jmp    short loc_gfc_loop_get_next_cluster
                             <1>
                             <1> floppy_drv_init:
                             <1>       ; 09/12/2017
                             <1>       ; 06/07/2016
                             <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 24/07/2011
                             <1>       ; 04/07/2009
                             <1>       ; INPUT ->
                             <1>       ;      DL = Drive number (0,1)
                             <1>       ; OUTPUT ->
                             <1>       ;      BL = drive name
                             <1>       ;      BH = drive number
                             <1>       ;      ESI = Logical DOS drv description table
                             <1>       ;      EAX = Volume serial number
                             <1>
00006B7E BE[F65C0000]        <1>       mov    esi, fd0_type ; 10/01/2016
00006B83 BF00010900          <1>       mov    edi, Logical_DOSDisks
00006B88 08D2                <1>       or     dl, dl
00006B8A 7407                <1>       jz     short loc_drv_init_fd0_fd1
00006B8C 81C700010000        <1>       add    edi, 100h
00006B92 46                  <1>       inc    esi ; fd1_type ; 10/01/2016
                             <1> loc_drv_init_fd0_fd1:
00006B93 C6477E00            <1>       mov    byte [edi+LD_MediaChanged], 0
00006B97 803E01              <1>       cmp    byte [esi], 1 ; type (>0 if it is existing)
                             <1>              ; 4 = 1.44 MB, 80 track, 3 1/2"
00006B9A 7221                <1>       jb     short read_fd_boot_sector_retn
00006B9C 885702              <1>       mov    [edi+LD_PhyDrvNo], dl
                             <1> read_fd_boot_sector:
00006B9F 30F6                <1>       xor    dh, dh
```

```
00006BA1 B904000000           <1>       mov    ecx, 4 ; Retry Count
                              <1> read_fd_boot_sector_again:
00006BA6 51                   <1>       push   ecx
                              <1>       ;mov   cx, 1
00006BA7 B101                 <1>       mov    cl, 1
00006BA9 66B80102             <1>       mov    ax, 0201h ; Read 1 sector
00006BAD BB[125F0100]         <1>       mov    ebx, DOSBootSectorBuff
00006BB2 E84FD6FFFF           <1>       call   int13h
00006BB7 59                   <1>       pop    ecx
00006BB8 7304                 <1>       jnc    short use_fd_boot_sector_params
00006BBA E2EA                 <1>       loop   read_fd_boot_sector_again
                              <1> 
                              <1> read_fd_boot_sector_stc_retn:
00006BBC F9                   <1>       stc
                              <1> read_fd_boot_sector_retn:
00006BBD C3                   <1>       retn
                              <1> 
                              <1> use_fd_boot_sector_params:
                              <1>       ;mov   esi, DOSBootSectorBuff
00006BBE 89DE                 <1>       mov    esi, ebx
00006BC0 6681BEFE01000055AA   <1>       cmp    word [esi+BS_Validation], 0AA55h
00006BC9 75F1                 <1>       jne    short read_fd_boot_sector_stc_retn
00006BCB 66817E035346         <1>         cmp     word [esi+bs_FS_Identifier], 'SF'
00006BD1 0F85A2000000         <1>         jne     use_fd_fatfs_boot_sector_params
                              <1>       ;
00006BD7 8A462D               <1>       mov    al, [esi+bs_FS_LBA_Ready]
00006BDA 884705               <1>       mov    [edi+LD_FS_LBAYes], al
                              <1>       ;
                              <1>       ; 03/01/2010 CHS -> DOS FAT/BPB compatibility fix
00006BDD 8A4608               <1>       mov    al, [esi+bs_FS_MediaAttrib]
00006BE0 884706               <1>       mov    [edi+LD_FS_MediaAttrib], al
                              <1>       ;
00006BE3 8A460A               <1>         mov  al, [esi+bs_FS_VersionMaj]
00006BE6 884707               <1>       mov    byte [edi+LD_FS_VersionMajor], al
00006BE9 668B4606             <1>       mov    ax, [esi+bs_FS_BytesPerSec]
00006BED 66894711             <1>       mov    [edi+LD_FS_BytesPerSec], ax
00006BF1 8A462E               <1>       mov    al, [esi+bs_FS_SecPerTrack]
00006BF4 28E4                 <1>       sub    ah, ah ; 09/12/2017
00006BF6 6689471E             <1>       mov    [edi+LD_FS_SecPerTrack], ax
00006BFA 8A462F               <1>       mov    al, [esi+bs_FS_Heads]
00006BFD 66894720             <1>       mov    [edi+LD_FS_NumHeads], ax
                              <1>       ;
00006C01 8B4628               <1>       mov    eax, [esi+bs_FS_UnDelDirD]
00006C04 894722               <1>       mov    [edi+LD_FS_UnDelDirD], eax
00006C07 8B4618               <1>       mov    eax, [esi+bs_FS_MATLocation]
00006C0A 89470C               <1>       mov    [edi+LD_FS_MATLocation], eax
00006C0D 8B461C               <1>       mov    eax, [esi+bs_FS_RootDirD]
00006C10 894708               <1>       mov    [edi+LD_FS_RootDirD], eax
00006C13 8B460C               <1>       mov    eax, [esi+bs_FS_BeginSector]
00006C16 89476C               <1>       mov    [edi+LD_FS_BeginSector], eax
00006C19 8B4610               <1>       mov    eax, [esi+bs_FS_VolumeSize]
00006C1C 894770               <1>       mov    [edi+LD_FS_VolumeSize], eax
                              <1>       ;
00006C1F 89FE                 <1>       mov    esi, edi
00006C21 8B460C               <1>       mov    eax, [esi+LD_FS_MATLocation]
                              <1>       ;add   eax, [edi+LD_FS_BeginSector]
                              <1> read_fd_MAT_sector_again:
                              <1>       ;mov   ebx, DOSBootSectorBuff
                              <1>       ;mov   ecx, 1
00006C24 B101                 <1>       mov    cl, 1
00006C26 E8B38B0000           <1>       call   chs_read
00006C2B 89DE                 <1>       mov    esi, ebx
00006C2D 7301                 <1>       jnc    short use_fdfs_mat_sector_params
                              <1>       ;jmp   short read_fd_boot_sector_retn
00006C2F C3                   <1>       retn
                              <1> use_fdfs_mat_sector_params:
00006C30 8B460C               <1>       mov    eax, [esi+FS_MAT_DATLocation]
00006C33 894714               <1>       mov    [edi+LD_FS_DATLocation], eax
00006C36 8B4610               <1>       mov    eax, [esi+FS_MAT_DATScount]
00006C39 894718               <1>       mov    [edi+LD_FS_DATSectors], eax
00006C3C 8B4714               <1>       mov    eax, [edi+FS_MAT_FreeSectors]
00006C3F 894774               <1>       mov    [edi+LD_FS_FreeSectors], eax
00006C42 8B4618               <1>       mov    eax, [esi+FS_MAT_FirstFreeSector]
00006C45 894778               <1>       mov    [edi+LD_FS_FirstFreeSector], eax
                              <1>       ;
00006C48 89FE                 <1>       mov    esi, edi
00006C4A 8B4608               <1>       mov    eax, [esi+LD_FS_RootDirD]
                              <1> read_fd_RDT_sector_again:
                              <1>       ;mov   ebx, DOSBootSectorBuff
                              <1>       ;mov   cx, 1
00006C4D B101                 <1>       mov    cl, 1
00006C4F E88A8B0000           <1>       call   chs_read
00006C54 89DE                 <1>       mov    esi, ebx
00006C56 7220                 <1>       jc     short read_fd_RDT_sector_retn
                              <1> use_fdfs_RDT_sector_params:
00006C58 8B461C               <1>       mov    eax, [esi+FS_RDT_VolumeSerialNo]
00006C5B 894728               <1>       mov    [edi+LD_FS_VolumeSerial], eax
00006C5E 57                   <1>       push   edi
                              <1>       ;mov   ecx, 16
00006C5F B110                 <1>       mov    cl, 16
00006C61 83C640               <1>       add    esi, FS_RDT_VolumeName
00006C64 83C72C               <1>       add    edi, LD_FS_VolumeName
00006C67 F3A5                 <1>       rep    movsd ; 64 bytes
00006C69 5E                   <1>       pop    esi
00006C6A C6460300             <1>       mov    byte [esi+LD_FATType], 0
00006C6E C64604A1             <1>       mov    byte [esi+LD_FSType], 0A1h
00006C72 E9A5000000           <1>         jmp     loc_cont_use_fd_boot_sector_params
                              <1> 
                              <1> read_fd_RDT_sector_stc_retn:
00006C77 F9                   <1>       stc
                              <1> read_fd_RDT_sector_retn:
00006C78 C3                   <1>       retn
                              <1> 
```

```
                             <1> use_fd_fatfs_boot_sector_params:
00006C79 807E2629            <1>       cmp    byte [esi+BS_BootSig], 29h
00006C7D 75F8                <1>       jne    short read_fd_RDT_sector_stc_retn
00006C7F 807E15F0            <1>       cmp    byte [esi+BPB_Media], 0F0h
00006C83 72F3                <1>       jb     short read_fd_RDT_sector_retn
00006C85 57                  <1>       push   edi
00006C86 83C706              <1>       add    edi, LD_BPB
                             <1>       ;mov   ecx, 16
00006C89 B110                <1>       mov    cl, 16
00006C8B F3A5                <1>       rep    movsd ; 64 bytes
00006C8D 5E                  <1>       pop    esi
00006C8E 31C0                <1>       xor    eax, eax
00006C90 89466C              <1>       mov    [esi+LD_StartSector], eax ; 0
00006C93 668B461C            <1>       mov    ax, [esi+LD_BPB+BPB_FATSz16]
00006C97 8A4E16              <1>       mov    cl, [esi+LD_BPB+BPB_NumFATs]
00006C9A F7E1                <1>       mul    ecx
                             <1>       ; edx = 0 !
00006C9C 668B5614            <1>       mov    dx, [esi+LD_BPB+BPB_RsvdSecCnt]
00006CA0 66895660            <1>       mov    [esi+LD_FATBegin], dx
                             <1>       ;add   eax, edx
00006CA4 6601D0              <1>       add    ax, dx
00006CA7 894664              <1>       mov    [esi+LD_ROOTBegin], eax
00006CAA 894668              <1>       mov    [esi+LD_DATABegin], eax
00006CAD 668B5617            <1>       mov    dx, [esi+LD_BPB+BPB_RootEntCnt]
                             <1>       ;;shl  edx, 5 ; * 32 (Size of a directory entry)
                             <1>       ;shl   dx, 5
                             <1>       ;;add  edx, 511
                             <1>       ;add   dx, 511
                             <1>       ;;shr  edx, 9 ; edx = ((edx*32)+511) / 512
                             <1>       ;shr   dx, 9
00006CB1 6683C20F            <1>       add    dx, 15 ; 06/07/2016 (+(512/32)-1)
00006CB5 66C1EA04            <1>       shr    dx, 4 ; / 16 (==16 entries per sector)
00006CB9 015668              <1>       add    [esi+LD_DATABegin], edx ; + rd sectors
                             <1>       ;movzx eax, word [esi+LD_BPB+BPB_TotalSec16]
00006CBC 668B4619            <1>       mov    ax, [esi+LD_BPB+BPB_TotalSec16]
00006CC0 894670              <1>       mov    [esi+LD_TotalSectors], eax
00006CC3 2B4668              <1>       sub    eax, [esi+LD_DATABegin]
                             <1>       ;movzx ecx, byte [esi+LD_BPB+BPB_SecPerClust]
00006CC6 8A4E13              <1>       mov    cl, [esi+LD_BPB+BPB_SecPerClust]
00006CC9 80F901              <1>       cmp    cl, 1
00006CCC 7605                <1>       jna    short save_fd_fatfs_cluster_count
                             <1>       ;sub   edx, edx
00006CCE 6629D2              <1>       sub    dx, dx ; 0
                             <1>       ;sub   dl, dl ; 06/07/2016
00006CD1 F7F1                <1>       div    ecx
                             <1> save_fd_fatfs_cluster_count:
00006CD3 894678              <1>       mov    [esi+LD_Clusters], eax
                             <1>
                             <1>       ; Maximum Valid Cluster Number = EAX +1
                             <1>       ; with 2 reserved clusters= EAX +2
                             <1>
                             <1> reset_FAT_buffer_decriptors:
00006CD6 29C0                <1>       sub    eax, eax ; 0
00006CD8 A2[16610100]        <1>       mov    [FAT_BuffValidData], al ; 0
00006CDD A2[17610100]        <1>       mov    [FAT_BuffDrvName], al ; 0
00006CE2 A3[1A610100]        <1>       mov    [FAT_BuffSector], eax ; 0
                             <1>
                             <1> read_fd_FAT_sectors:
00006CE7 BB001C0900          <1>       mov    ebx, FAT_Buffer
00006CEC 668B4614            <1>       mov    ax, [esi+LD_BPB+BPB_RsvdSecCnt]
                             <1>       ;mov   ecx, 3
00006CF0 B103                <1>       mov    cl, 3 ; 3 sectors
00006CF2 E8E78A0000          <1>       call   chs_read
00006CF7 7240                <1>       jc     short read_fd_FAT_sectors_retn
                             <1> use_fd_FAT_sectors:
00006CF9 8A4602              <1>       mov    al, [esi+LD_PhyDrvNo]
00006CFC 0441                <1>       add    al, 'A'
00006CFE A2[17610100]        <1>       mov    [FAT_BuffDrvName], al
00006D03 C605[16610100]01    <1>       mov    byte [FAT_BuffValidData], 1
00006D0A E82B000000          <1>       call   fd_init_calculate_free_clusters
00006D0F 7228                <1>       jc     short read_fd_FAT_sectors_retn
                             <1>
                             <1> loc_use_fd_boot_sector_params_FAT:
00006D11 C6460301            <1>       mov    byte [esi+LD_FATType], 1 ; FAT 12
00006D15 C6460401            <1>       mov    byte [esi+LD_FSType], 1
00006D19 8B462D              <1>         mov     eax, [esi+LD_BPB+VolumeID]
                             <1> loc_cont_use_fd_boot_sector_params:
00006D1C 8A7E02              <1>       mov    bh, [esi+LD_PhyDrvNo]
00006D1F 887E7D              <1>       mov    [esi+LD_DParamEntry], bh
00006D22 88FB                <1>       mov    bl, bh
00006D24 80C341              <1>       add    bl, 'A'
00006D27 881E                <1>       mov    byte [esi+LD_Name], bl
00006D29 C6460101            <1>       mov    byte [esi+LD_DiskType], 1
00006D2D C6460500            <1>       mov    byte [esi+LD_LBAYes], 0
00006D31 C6467C00            <1>       mov    byte [esi+LD_PartitionEntry], 0
00006D35 C6467E06            <1>       mov    byte [esi+LD_MediaChanged], 6 ; Volume Name Reset
                             <1>
                             <1> read_fd_FAT_sectors_retn:
00006D39 C3                  <1>       retn
                             <1>
                             <1> fd_init_calculate_free_clusters:
                             <1>       ; 09/12/2017
                             <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 04/07/2009
                             <1>       ; INPUT ->
                             <1>       ;     ESI = Logical DOS drive description table address
                             <1>       ; OUTPUT ->
                             <1>       ;    [ESI+LD_FreeSectors] will be set
                             <1>
00006D3A 29C0                <1>       sub    eax, eax
00006D3C 894674              <1>       mov    [esi+LD_FreeSectors], eax ; 0
00006D3F B002                <1>       mov    al, 2 ; eax = 2
                             <1>
                             <1> fd_init_loop_get_next_cluster:
00006D41 E830000000          <1>       call   fd_init_get_next_cluster
00006D46 722D                <1>       jc     short fd_init_calculate_free_clusters_retn
                             <1> 
                             <1> fd_init_free_fat_clusters:
                             <1>       ;cmp   eax, 0
                             <1>       ;ja    short fd_init_pass_inc_free_cluster_count
                             <1>       ;and   eax, eax
                             <1>       ;jnz   short fd_init_pass_inc_free_cluster_count
00006D48 6621C0              <1>       and    ax, ax
00006D4B 7504                <1>       jnz    short fd_init_pass_inc_free_cluster_count
                             <1>       ;inc   dword [esi+LD_FreeSectors]
00006D4D 66FF4674            <1>         inc  word [esi+LD_FreeSectors]
                             <1> 
                             <1> fd_init_pass_inc_free_cluster_count:
                             <1>       ;mov   eax, [FAT_CurrentCluster]
00006D51 66A1[12610100]      <1>       mov    ax, [FAT_CurrentCluster]
                             <1>       ;cmp   eax, [esi+LD_Clusters]
00006D57 663B4678            <1>       cmp    ax, [esi+LD_Clusters]
00006D5B 7704                <1>       ja     short short retn_from_fd_init_calculate_free_clusters
                             <1>       ;inc   eax
00006D5D 6640                <1>       inc    ax
00006D5F EBE0                <1>       jmp    short fd_init_loop_get_next_cluster
                             <1> 
                             <1> retn_from_fd_init_calculate_free_clusters:
00006D61 8A4613              <1>       mov    al, [esi+LD_BPB+BPB_SecPerClust]
00006D64 3C01                <1>       cmp    al, 1
00006D66 760D                <1>       jna    short fd_init_calculate_free_clusters_retn
                             <1>       ;movzx eax, al
00006D68 30E4                <1>       xor    ah, ah ; 09/12/2017
                             <1>       ;mov   ecx, [esi+LD_FreeSectors]
00006D6A 668B4E74            <1>       mov    cx, [esi+LD_FreeSectors] ; Count of free clusters
                             <1>       ;mul   ecx
00006D6E 66F7E1              <1>       mul    cx
                             <1>       ;mov   [esi+LD_FreeSectors], eax
00006D71 66894674            <1>       mov    [esi+LD_FreeSectors], ax
                             <1> fd_init_calculate_free_clusters_retn:
00006D75 C3                  <1>       retn
                             <1> 
                             <1> fd_init_get_next_cluster:
                             <1>       ; 04/02/2016
                             <1>       ; 02/02/2016
                             <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 04/07/2009
                             <1>       ; INPUT ->
                             <1>       ;    EAX = Current cluster
                             <1>       ;    ESI = Logical DOS drive description table address
                             <1>       ;    EDX = 0
                             <1>       ; OUTPUT ->
                             <1>       ;    EAX = Next cluster
                             <1> 
00006D76 A3[12610100]        <1>       mov    [FAT_CurrentCluster], eax
                             <1> fd_init_get_next_cluster_readnext:
00006D7B 29D2                <1>       sub    edx, edx ; 0
00006D7D BB00040000          <1>       mov    ebx, 1024 ; 400h
00006D82 F7F3                <1>       div    ebx
                             <1>       ; EAX = Count of 3 FAT sectors
                             <1>       ; EDX = Buffer entry index
00006D84 89C1                <1>       mov    ecx, eax
                             <1>       ;mov   eax, 3
00006D86 B003                <1>       mov    al, 3
00006D88 F7E2                <1>       mul    edx ; Multiply by 3
00006D8A 66D1E8              <1>       shr    ax, 1 ; Divide by 2
00006D8D 89C3                <1>       mov    ebx, eax ; Buffer byte offset
00006D8F 81C3001C0900        <1>       add    ebx, FAT_Buffer
00006D95 89C8                <1>       mov    eax, ecx
                             <1>       ;mov   edx, 3
00006D97 66BA0300            <1>       mov    dx, 3
00006D9B F7E2                <1>       mul    edx
                             <1>       ; EAX = FAT Beginning Sector
                             <1>       ; EDX = 0
00006D9D 8A0E                <1>       mov    cl, [esi+LD_Name]
                             <1>       ;cmp   byte [FAT_BuffValidData], 0
                             <1>       ;jna   short fd_init_load_FAT_sectors0
00006D9F 3A0D[17610100]      <1>       cmp    cl, [FAT_BuffDrvName]
00006DA5 751E                <1>       jne    short fd_init_load_FAT_sectors0
00006DA7 3B05[1A610100]      <1>       cmp    eax, [FAT_BuffSector]
00006DAD 751C                <1>       jne    short fd_init_load_FAT_sectors1
                             <1>       ;mov   eax, [FAT_CurrentCluster]
00006DAF A0[12610100]        <1>       mov    al, [FAT_CurrentCluster]
                             <1>       ;shr   eax, 1
00006DB4 D0E8                <1>       shr    al, 1
00006DB6 668B03              <1>       mov    ax, [ebx]
00006DB9 7306                <1>       jnc    short fd_init_gnc_even
00006DBB 66C1E804            <1>       shr    ax, 4
                             <1> fd_init_gnc_clc_retn:
00006DBF F8                  <1>       clc
00006DC0 C3                  <1>       retn
                             <1> 
                             <1> fd_init_gnc_even:
00006DC1 80E40F              <1>       and    ah, 0Fh
00006DC4 C3                  <1>       retn
                             <1> 
                             <1> fd_init_load_FAT_sectors0:
00006DC5 880D[17610100]      <1>       mov    [FAT_BuffDrvName], cl
                             <1> fd_init_load_FAT_sectors1:
00006DCB C605[16610100]00    <1>       mov    byte [FAT_BuffValidData], 0
00006DD2 A3[1A610100]        <1>       mov    [FAT_BuffSector], eax
00006DD7 034660              <1>       add    eax, [esi+LD_FATBegin]
00006DDA BB001C0900          <1>       mov    ebx, FAT_Buffer
                             <1>       ;movzx ecx, word [esi+LD_BPB+BPB_FATSz16]
00006DDF 668B4E1C            <1>       mov    cx, [esi+LD_BPB+BPB_FATSz16]
00006DE3 662B0D[1A610100]    <1>       sub    cx, [FAT_BuffSector]
```

```
                             <1>         ;cmp ecx, 3
00006DEA 6683F903            <1>       cmp   cx, 3
00006DEE 7605                <1>       jna   short fdinit_pass_fix_sector_count_3
                             <1>       ;mov  ecx, 3
00006DF0 B903000000          <1>       mov   ecx, 3
                             <1> fdinit_pass_fix_sector_count_3:
00006DF5 E8E4890000          <1>       call  chs_read
00006DFA 730D                <1>       jnc   short fd_init_FAT_sectors_no_load_error
00006DFC C605[16610100]00    <1>       mov   byte [FAT_BuffValidData], 0
                             <1>             ; Drv not ready or read Error !
00006E03 B80F000000          <1>       mov   eax, ERR_DRV_NOT_RDY ; 15
                             <1>       ;xor  edx, edx
00006E08 C3                  <1>       retn
                             <1>
                             <1> fd_init_FAT_sectors_no_load_error:
00006E09 C605[16610100]01    <1>       mov   byte [FAT_BuffValidData], 1
00006E10 A1[12610100]        <1>       mov   eax, [FAT_CurrentCluster]
00006E15 E961FFFFFF          <1>         jmp     fd_init_get_next_cluster_readnext
                             <1>
                             <1> get_FAT_volume_name:
                             <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 12/09/2009
                             <1>       ; INPUT ->
                             <1>       ;     BH = Logical DOS drive number (0,1,2,3,4 ...)
                             <1>       ;       BL = 0
                             <1>       ; OUTPUT ->
                             <1>       ;     CF = 0 -> ESI = Volume name address
                             <1>       ;     CF = 1 -> Root volume name not found
                             <1>
                             <1>       ;mov  ah, 0FFh
                             <1>       ;mov  al, [Last_Dos_DiskNo]
                             <1>       ;cmp  al, bh
                             <1>       ;jb     short loc_gfvn_dir_load_err
                             <1>
00006E1A 89DE                <1>       mov   esi, ebx
00006E1C 81E600FF0000        <1>       and   esi, 0FF00h ; esi = bh
00006E22 81C600010900        <1>       add   esi, Logical_DOSDisks
00006E28 8A06                <1>       mov     al, [esi+LD_Name]
00006E2A 8A6603              <1>       mov     ah, [esi+LD_FATType]
00006E2D 80FC01              <1>       cmp     ah, 1
00006E30 7210                <1>       jb    short loc_gfvn_dir_load_err
00006E32 3C41                <1>       cmp   al, 'A'
00006E34 720C                <1>       jb      short loc_gfvn_dir_load_err
00006E36 80FC02              <1>       cmp   ah, 2
00006E39 7708                <1>       ja      short get_FAT32_root_cluster
                             <1>
00006E3B E8634E0000          <1>       call    load_FAT_root_directory
00006E40 730B                <1>       jnc     short loc_get_volume_name
                             <1>
                             <1> loc_gfvn_dir_load_err:
00006E42 C3                  <1>       retn
                             <1>
                             <1> get_FAT32_root_cluster:
00006E43 8B4632              <1>       mov   eax, [esi+LD_BPB+BPB_RootClus]
00006E46 E8E34E0000          <1>       call    load_FAT_sub_directory
00006E4B 7224                <1>       jc    short loc_get_volume_name_retn
                             <1>
                             <1> loc_get_volume_name:
00006E4D BE00000800          <1>         mov     esi, Directory_Buffer
00006E52 6631C9              <1>       xor   cx, cx ; 0
                             <1> check_root_volume_name:
00006E55 8A06                <1>       mov   al, [esi]
00006E57 08C0                <1>       or      al, al
00006E59 7416                <1>       jz      short loc_get_volume_name_retn
00006E5B 807E0B08            <1>       cmp     byte [esi+0Bh], 08h
00006E5F 7410                <1>       je      short loc_get_volume_name_retn
00006E61 663B0D[2B610100]    <1>       cmp     cx, [DirBuff_LastEntry]
00006E68 7308                <1>       jnb     short pass_check_root_volume_name
00006E6A 6641                <1>       inc     cx
00006E6C 83C620              <1>       add     esi, 32
00006E6F EBE4                <1>       jmp     short check_root_volume_name
                             <1>
                             <1> loc_get_volume_name_retn:
00006E71 C3                  <1>       retn
                             <1>
                             <1> pass_check_root_volume_name:
00006E72 803D[27610100]03    <1>       cmp   byte [DirBuff_FATType], 3
00006E79 7230                <1>       jb    short loc_get_volume_name_retn_xor
                             <1>
00006E7B BB001C0900          <1>       mov   ebx, FAT_Buffer
00006E80 BE00010900          <1>       mov   esi, Logical_DOSDisks
00006E85 31C0                <1>       xor   eax, eax
00006E87 8A25[26610100]      <1>       mov   ah, [DirBuff_DRV]
00006E8D 80EC41              <1>       sub   ah, 'A'
00006E90 01C6                <1>       add   esi, eax
00006E92 A1[2D610100]        <1>       mov   eax, [DirBuff_Cluster]
00006E97 E8AC4C0000          <1>       call  get_next_cluster
00006E9C 7305                <1>       jnc   short loc_gfvn_load_FAT32_dir_cluster
                             <1>
00006E9E 83F801              <1>       cmp     eax, 1
00006EA1 F5                  <1>       cmc
00006EA2 C3                  <1>       retn
                             <1>
                             <1> loc_gfvn_load_FAT32_dir_cluster:
00006EA3 E8864E0000          <1>       call  load_FAT_sub_directory
00006EA8 73A3                <1>       jnc   short loc_get_volume_name
00006EAA C3                  <1>       retn
                             <1>
                             <1> loc_get_volume_name_retn_xor:
00006EAB 31C0                <1>       xor   eax, eax
00006EAD C3                  <1>       retn
                             <1>
                             <1> get_media_change_status:
```

```
                         <1>       ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
                         <1>       ; 09/09/2009
                         <1>       ; INPUT:
                         <1>       ;     DL = Drive number (physical)
                         <1>       ; OUTPUT: clc & AH = 6 media changed
                         <1>       ;     clc & AH = 0 media not changed
                         <1>       ;     stc -> Drive not ready or an error
                         <1>
00006EAE B416            <1>       mov    ah, 16h
00006EB0 E851D3FFFF      <1>       call   int13h
00006EB5 80FC06          <1>       cmp    ah, 06h
00006EB8 7405            <1>       je     short loc_gmc_status_retn
00006EBA 08E4            <1>       or     ah, ah
00006EBC 7401            <1>       jz     short loc_gmc_status_retn
                         <1> loc_gmc_status_stc_retn:
00006EBE F9              <1>       stc
                         <1> loc_gmc_status_retn:
00006EBF C3              <1>       retn
                             %include 'trdosk3.s' ; 06/01/2016
                         <1> ; ****************************************************************************
                         <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - MAIN PROGRAM : trdosk3.s
                         <1> ; ----------------------------------------------------------------------------
                         <1> ; Last Update: 31/12/2017
                         <1> ; ----------------------------------------------------------------------------
                         <1> ; Beginning: 06/01/2016
                         <1> ; ----------------------------------------------------------------------------
                         <1> ; Assembler: NASM version 2.11 (trdos386.s)
                         <1> ; ----------------------------------------------------------------------------
                         <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
                         <1> ; MAINPROG.ASM (09/11/2011)
                         <1> ; ****************************************************************************
                         <1> ; MAINPROG.ASM [ TRDOS KERNEL - COMMAND EXECUTER SECTION - MAIN PROGRAM ]
                         <1> ; (c) 2004-2011  Erdogan TAN  [ 17/01/2004 ]  Last Update: 09/11/2011
                         <1> ; CMD_INTR.ASM [ TRDOS Command Interpreter Procedure ] Last Update: 09/11/2011
                         <1> ; DIR.ASM [ DIRECTORY FUNCTIONS ] Last Update: 09/10/2011
                         <1> ; FILE.ASM [ FILE FUNCTIONS ] Last Update: 09/10/2011
                         <1>
                         <1> change_current_drive:
                         <1>       ; 16/10/2016
                         <1>       ; 02/02/2016
                         <1>       ; 15/01/2016 (TRDOS 386 = TRDOS v2.0)
                         <1>       ; 18/08/2011
                         <1>       ; 09/09/2009
                         <1>       ; INPUT:
                         <1>       ;   DL = Logical DOS Drive Number
                         <1>       ; OUTPUT:
                         <1>       ;  cf=1 -> Not successful
                         <1>       ;   EAX = Error code
                         <1>       ;  cf=0 ->
                         <1>       ;   EAX = 0 (successful)
                         <1>
00006EC0 31DB            <1>       xor    ebx, ebx
00006EC2 88D7            <1>       mov    bh, dl
                         <1>
                         <1>       ;cmp   dl, 1
                         <1>       ;jna   short loc_ccdrv_initial_media_change_check
                         <1>       ;cmp   bh, [Last_Dos_DiskNo]
                         <1>       ;ja    short loc_ccdrv_drive_not_ready_err
                         <1>
                         <1> loc_ccdrv_initial_media_change_check:
00006EC4 BE00010900      <1>       mov    esi, Logical_DOSDisks
00006EC9 01DE            <1>       add    esi, ebx
                         <1> loc_ccdrv_dos_drive_name_check:
00006ECB 80FA02          <1>       cmp    dl, 2
00006ECE 720F            <1>       jb     short loc_ccdrv_dos_drive_name_check_ok
                         <1>
00006ED0 8A06            <1>       mov    al, [esi+LD_Name]
00006ED2 2C41            <1>       sub    al, 'A'
00006ED4 38D0            <1>       cmp    al, dl
00006ED6 7407            <1>       je     short loc_ccdrv_dos_drive_name_check_ok
                         <1>
                         <1> loc_ccdrv_drive_not_ready_err:
                         <1>       ; 16/10/2016 (15h -> 15)
00006ED8 B80F000000      <1>       mov    eax, 15 ; Drive not ready
                         <1> loc_change_current_drive_stc_retn:
00006EDD F9              <1>       stc
00006EDE C3              <1>       retn
                         <1>
                         <1> loc_ccdrv_dos_drive_name_check_ok:
00006EDF 8A667E          <1>       mov    ah, [esi+LD_MediaChanged]
00006EE2 80FC06          <1>       cmp    ah, 6  ; VOLUME NAME CHECK/MOVE SIGN
00006EE5 7455            <1>       je     short loc_ccdrv_get_FAT_volume_name_0
                         <1>
00006EE7 80FA01          <1>       cmp    dl, 1
00006EEA 777D            <1>       ja     short loc_gmcs_init_drv_hd
                         <1>
                         <1> loc_gmcs_init_drv_fd:
00006EEC 08E4            <1>       or     ah, ah
                         <1>       ; AH = 1 is initialization sign (invalid_fd_parameter)
00006EEE 7517            <1>       jnz    short loc_ccdrv_call_fd_init
                         <1>
00006EF0 E8B9FFFFFF      <1>       call   get_media_change_status
00006EF5 72E1            <1>       jc     short loc_ccdrv_drive_not_ready_err
                         <1>
00006EF7 20E4            <1>       and    ah, ah
00006EF9 7476            <1>       jz     short loc_change_current_drv3
                         <1>
00006EFB 80F406          <1>       xor    ah, 6
00006EFE 75D8            <1>       jnz    short loc_ccdrv_drive_not_ready_err
                         <1>
                         <1> loc_ccdrv_call_fd_init_check_vol_id:
00006F00 E8440A0000      <1>       call   get_volume_serial_number
00006F05 730D            <1>       jnc    short loc_ccdrv_check_vol_serial
```

```
                           <1> 
                           <1> loc_ccdrv_call_fd_init:
00006F07 E872FCFFFF        <1>       call   floppy_drv_init
00006F0C 731A              <1>       jnc    short loc_reset_drv_fd_current_dir
                           <1> 
                           <1> loc_ccdrv_fdinit_fail_retn:
                           <1>       ; 16/10/2016
00006F0E B80F000000        <1>       mov    eax, 15 ; Drive not ready
00006F13 C3                <1>       retn
                           <1> 
                           <1> loc_ccdrv_check_vol_serial:
00006F14 A3[F4580100]      <1>       mov    [Current_VolSerial], eax
                           <1>       ;mov   dl, bh
00006F19 E860FCFFFF        <1>       call   floppy_drv_init
00006F1E 72EE              <1>       jc     short loc_ccdrv_fdinit_fail_retn
                           <1> 
00006F20 3B05[F4580100]    <1>       cmp    eax, [Current_VolSerial]
00006F26 7445              <1>       je     short loc_change_current_drv2
                           <1> 
                           <1> loc_reset_drv_fd_current_dir:
00006F28 31C0              <1>       xor    eax, eax
00006F2A 88467F            <1>         mov  [esi+LD_CDirLevel], al
00006F2D 89F7              <1>       mov    edi, esi
00006F2F 81C780000000      <1>       add    edi, LD_CurrentDirectory
00006F35 B920000000        <1>       mov    ecx, 32
00006F3A F3AB              <1>       rep    stosd
                           <1> 
                           <1> loc_ccdrv_get_FAT_volume_name_0:
00006F3C 8A4603            <1>       mov    al, [esi+LD_FATType]
00006F3F 08C0              <1>       or     al, al
00006F41 742A              <1>       jz     short loc_change_current_drv2
                           <1> 
00006F43 56                <1>       push   esi
00006F44 3C02              <1>       cmp    al, 2
00006F46 7705              <1>       ja     short loc_ccdrv_get_FAT32_vol_name
                           <1> 
                           <1> loc_ccdrv_get_FAT2_16_vol_name:
00006F48 83C631            <1>       add    esi, LD_BPB + VolumeLabel
00006F4B EB03              <1>       jmp    short loc_ccdrv_get_FAT_volume_name_1
                           <1> 
                           <1> loc_ccdrv_get_FAT32_vol_name:
00006F4D 83C64D            <1>       add    esi, LD_BPB + FAT32_VolLab
                           <1> loc_ccdrv_get_FAT_volume_name_1:
00006F50 53                <1>       push   ebx
00006F51 56                <1>       push   esi
00006F52 E8C3FEFFFF        <1>       call   get_FAT_volume_name
00006F57 5F                <1>       pop    edi
00006F58 5B                <1>       pop    ebx
                           <1>       ; BL = 0
00006F59 720B              <1>       jc     short loc_change_current_drv1
00006F5B 20C0              <1>       and    al, al
00006F5D 7407              <1>       jz     short loc_change_current_drv1
                           <1> 
                           <1> loc_ccdrv_move_FAT_volume_name:
00006F5F B90B000000        <1>       mov    ecx, 11
00006F64 F3A4              <1>       rep    movsb
                           <1> 
                           <1> loc_change_current_drv1:
00006F66 5E                <1>       pop    esi
00006F67 EB04              <1>       jmp    short loc_change_current_drv2
                           <1> 
                           <1> loc_gmcs_init_drv_hd:
00006F69 08E4              <1>       or     ah, ah
00006F6B 7404              <1>       jz     short loc_change_current_drv3
                           <1>       ; BL = 0, BH = Logical DOS drive number
                           <1> loc_change_current_drv2:
00006F6D C6467E00          <1>       mov    byte [esi+LD_MediaChanged], 0
                           <1> loc_change_current_drv3:
00006F71 883D[FE580100]    <1>       mov    [Current_Drv], bh
                           <1> 
                           <1>       ;call  restore_current_directory
                           <1>       ;retn
                           <1> 
                           <1> restore_current_directory:
                           <1>       ; 11/02/2016
                           <1>       ; 15/01/2016 (TRDOS 386 = TRDOS v2.0)
                           <1>       ; 25/01/2010
                           <1>       ; 12/10/2009
                           <1>       ;
                           <1>       ; INPUT:
                           <1>       ;   ESI = Logical DOS Drive Description Table
                           <1>       ;
                           <1>       ; OUTPUT:
                           <1>       ;   ESI = Logical DOS Drive Description Table
                           <1>       ;   EDI = offset Current_Dir_Drv
                           <1> 
00006F77 8A4603            <1>       mov    al, [esi+LD_FATType]
00006F7A A2[FD580100]      <1>       mov    [Current_FATType], al
                           <1> 
00006F7F 8A26              <1>       mov    ah, [esi+LD_Name]
00006F81 8825[FF580100]    <1>       mov    [Current_Dir_Drv], ah
                           <1> 
00006F87 20C0              <1>       and    al, al
00006F89 741D              <1>       jz     short loc_restore_FS_current_directory
                           <1> 
                           <1> loc_restore_FAT_current_directory:
00006F8B 8A667F            <1>       mov    ah, [esi+LD_CDirLevel]
00006F8E 8825[FC580100]    <1>       mov    [Current_Dir_Level], ah
00006F94 08E4              <1>       or     ah, ah
00006F96 7416              <1>         jz   short loc_ccdrv_reset_cdir_FAT_12_16_32_fcluster
                           <1> 
00006F98 0FB6D4            <1>       movzx  edx, ah
00006F9B C0E204            <1>       shl    dl, 4 ; * 16
```

```
00006F9E 01F2                <1>         add  edx, esi
00006FA0 8B828C000000        <1>       mov    eax, [edx+LD_CurrentDirectory+12]
00006FA6 EB2C                <1>       jmp    short loc_ccdrv_reset_cdir_FAT_fcluster
                             <1> 
                             <1> loc_restore_FS_current_directory:
00006FA8 E8BC4D0000          <1>       call   load_current_FS_directory
00006FAD C3                  <1>       retn
                             <1> 
                             <1> loc_ccdrv_reset_cdir_FAT_12_16_32_fcluster:
00006FAE 3C03                <1>       cmp    al, 3
00006FB0 7205                <1>       jb     short loc_ccdrv_reset_cdir_FAT_12_16_fcluster
                             <1> loc_ccdrv_reset_cdir_FAT32_fcluster:
00006FB2 8B4632              <1>       mov    eax, [esi+LD_BPB+FAT32_RootFClust]
00006FB5 EB04                <1>       jmp    short loc_ccdrv_check_rootdir_sign
                             <1> loc_ccdrv_reset_cdir_FAT_12_16_fcluster:
00006FB7 30C0                <1>       xor    al, al  ; xor eax, eax
00006FB9 31D2                <1>       xor    edx, edx
                             <1> loc_ccdrv_check_rootdir_sign:
00006FBB 80BE8000000000      <1>       cmp    byte [esi+LD_CurrentDirectory], 0
00006FC2 7510                <1>       jne    short loc_ccdrv_reset_cdir_FAT_fcluster
                             <1> loc_ccdrv_set_rootdir_FAT_fcluster:
00006FC4 89868C000000        <1>         mov     [esi+LD_CurrentDirectory+12], eax
00006FCA C78680000000524F4F- <1>       mov    dword [esi+LD_CurrentDirectory], 'ROOT'
00006FD3 54                  <1>
                             <1> 
                             <1> loc_ccdrv_reset_cdir_FAT_fcluster:
00006FD4 A3[F8580100]        <1>       mov    [Current_Dir_FCluster], eax
                             <1> 
00006FD9 BF[5F610100]        <1>       mov    edi, PATH_Array
00006FDE 89F2                <1>       mov    edx, esi
00006FE0 81C680000000        <1>       add    esi, LD_CurrentDirectory
00006FE6 B920000000          <1>       mov    ecx, 32
00006FEB F3A5                <1>       rep    movsd
                             <1> 
00006FED E84C2D0000          <1>       call   change_prompt_dir_string
                             <1> 
00006FF2 89D6                <1>       mov    esi, edx
                             <1> 
00006FF4 29C0                <1>         sub  eax, eax
                             <1>        ;sub  edx, edx
00006FF6 BF[FF580100]        <1>       mov    edi, Current_Dir_Drv
                             <1> 
00006FFB A2[D30C0100]        <1>       mov    [Restore_CDIR], al ; 0
00007000 C3                  <1>       retn
                             <1> 
                             <1> dos_prompt:
                             <1>       ; 06/05/2016
                             <1>       ; 30/01/2016
                             <1>       ; 29/01/2016
                             <1>       ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 15/09/2011
                             <1>       ; 13/09/2009
                             <1>       ; 2004-2005
                             <1> 
                             <1>       ; 06/05/2016
00007001 C705[BC650100]-     <1>       mov    dword [mainprog_return_addr], return_from_cmd_interpreter
00007007 [B5700000]          <1>
                             <1> 
                             <1> loc_TRDOS_prompt:
0000700B BF[FE590100]        <1>       mov    edi, TextBuffer
00007010 C6075B              <1>       mov    byte [edi], "["
00007013 47                  <1>       inc    edi
00007014 BE[260D0100]        <1>       mov    esi, TRDOSPromptLabel
                             <1> get_next_prompt_label_char:
00007019 803E20              <1>       cmp    byte [esi], 20h
0000701C 7203                <1>       jb     short pass_prompt_label
0000701E A4                  <1>       movsb
0000701F EBF8                <1>       jmp    short get_next_prompt_label_char
                             <1> pass_prompt_label:
00007021 C6075D              <1>       mov    byte [edi], "]"
00007024 47                  <1>       inc    edi
00007025 C60720              <1>       mov    byte [edi], 20h
00007028 47                  <1>       inc    edi
00007029 BE[FF580100]        <1>       mov    esi, Current_Dir_Drv
0000702E 66A5                <1>       movsw
00007030 A4                  <1>       movsb
                             <1> loc_prompt_current_directory:
00007031 803E20              <1>       cmp    byte [esi], 20h
00007034 7203                <1>       jb     short pass_prompt_current_directory
00007036 A4                  <1>       movsb
00007037 EBF8                <1>       jmp    short loc_prompt_current_directory
                             <1> pass_prompt_current_directory:
00007039 C6073E              <1>       mov    byte [edi], '>'
0000703C 47                  <1>       inc    edi
0000703D C60700              <1>       mov    byte [edi], 0
00007040 BE[FE590100]        <1>       mov    esi, TextBuffer
00007045 E813F3FFFF          <1>       call   print_msg
                             <1> 
                             <1>       ;sub   bh, bh ; video page = 0
                             <1>       ;call  get_cpos ; get cursor position
0000704A 668B15[56580100]    <1>       mov    dx, [CURSOR_POSN] ; video page 0
00007051 8815[5E590100]      <1>       mov    [CursorColumn], dl
                             <1> 
                             <1>       ; 30/01/2016 (to show cursor on the row, again)
                             <1>       ; (Initial color attributes of video page 0 is 0)
                             <1>       ; (see: 'StartPMP' in trdos386.s)
                             <1>       ;
                             <1>       ;mov   edi, 0B8000h ; start of video page 0
                             <1>       ;movzx ecx, dl ; column
                             <1>       ;mov   al, 80
                             <1>       ;mul   dh
                             <1>       ;add   ax, cx
                             <1>       ;shl   ax, 1 ; character + attribute
                             <1>       ;add   di, ax ; (2*80*row) + (2*column)
                             <1>       ;neg   cl
                             <1>       ;add   cl, 80
                             <1>       ;mov   ax, 700h ;  ah = 7 (color attribute)
                             <1>       ;rep   stosw
                             <1>
                             <1> loc_rw_char:
00007057 E899000000          <1>       call   rw_char
                             <1> loc_move_command:
0000705C BE[AE590100]        <1>       mov    esi, CommandBuffer
00007061 89F7                <1>       mov    edi, esi
00007063 31C9                <1>       xor    ecx, ecx
                             <1> first_command_char:
00007065 AC                  <1>       lodsb
00007066 3C20                <1>       cmp    al, 20h
00007068 772E                <1>       ja     short pass_space_control
0000706A 7241                <1>       jb     short loc_move_cmd_arguments_ok
0000706C 81FE[FD590100]      <1>       cmp    esi, CommandBuffer + 79
00007072 72F1                <1>       jb     short first_command_char
00007074 EB37                <1>       jmp    short loc_move_cmd_arguments_ok
                             <1>
                             <1> next_command_char:
00007076 AC                  <1>       lodsb
00007077 3C20                <1>       cmp    al, 20h
00007079 771D                <1>       ja     short pass_space_control
0000707B 7230                <1>       jb     short loc_move_cmd_arguments_ok
                             <1>
                             <1> loc_1st_cmd_arg: ; 30/01/2016
0000707D AC                  <1>       lodsb
0000707E 3C20                <1>       cmp    al, 20h
00007080 74FB                <1>       je     short loc_1st_cmd_arg
00007082 7229                <1>       jb     short loc_move_cmd_arguments_ok
                             <1>
00007084 C60700              <1>         mov     byte [edi], 0
00007087 47                  <1>       inc    edi
                             <1>
                             <1> loc_move_cmd_arguments:
00007088 AA                  <1>       stosb
00007089 81FE[FD590100]      <1>       cmp    esi, CommandBuffer + 79
0000708F 731C                <1>       jnb    short loc_move_cmd_arguments_ok
00007091 AC                  <1>         lodsb
00007092 3C20                <1>       cmp    al, 20h
00007094 73F2                <1>       jnb    short loc_move_cmd_arguments
00007096 EB15                <1>       jmp    short loc_move_cmd_arguments_ok
                             <1>
                             <1> pass_space_control:
00007098 3C61                <1>       cmp    al, 61h
0000709A 7206                <1>       jb     short pass_capitalize
0000709C 3C7A                <1>       cmp    al, 7Ah
0000709E 7702                <1>       ja     short pass_capitalize
000070A0 24DF                <1>       and    al, 0DFh
                             <1> pass_capitalize:
000070A2 AA                  <1>       stosb
000070A3 FEC1                <1>       inc     cl
000070A5 81FE[FD590100]      <1>         cmp     esi, CommandBuffer + 79
000070AB 72C9                <1>       jb      short next_command_char
                             <1>
                             <1> loc_move_cmd_arguments_ok:
000070AD C60700              <1>         mov     byte [edi], 0
                             <1>
                             <1> call_command_interpreter:
000070B0 E8CF080000          <1>       call    command_interpreter
                             <1>
                             <1> return_from_cmd_interpreter:
000070B5 B950000000          <1>         mov ecx, 80
                             <1>       ;mov   cx, 80
000070BA BF[AE590100]        <1>       mov    edi, CommandBuffer
000070BF 30C0                <1>       xor    al, al
000070C1 F3AA                <1>       rep    stosb
                             <1>       ;cmp   byte [Program_Exit], 0
                             <1>       ;ja    short loc_terminate_trdos
                             <1>
                             <1>       ; 16/01/2016
000070C3 803D[C25E0000]03    <1>       cmp    byte [CRT_MODE], 3 ; 80*25 color
000070CA 741D                <1>       je     short pass_set_txt_mode
                             <1>
000070CC E892A4FFFF          <1>       call   set_txt_mode ; set vide mode to 03h
                             <1>       ; 07/01/2017
000070D1 30C0                <1>       xor    al, al
                             <1>
                             <1> loc_check_active_page:
                             <1>       ;xor   al, al
000070D3 3805[66580100]      <1>       cmp    [ACTIVE_PAGE], al ; 0
000070D9 0F842CFFFFFF        <1>         je      loc_TRDOS_prompt
                             <1>       ; AL = 0 = video page 0
000070DF E898A8FFFF          <1>       call   set_active_page
000070E4 E922FFFFFF          <1>         jmp     loc_TRDOS_prompt ; infinitive loop
                             <1>
                             <1> pass_set_txt_mode:
000070E9 BE[6F190100]        <1>       mov    esi, nextline
000070EE E86AF2FFFF          <1>       call   print_msg
000070F3 EBDE                <1>       jmp     short loc_check_active_page
                             <1>
                             <1> rw_char:
                             <1>       ; 13/05/2016
                             <1>       ; 30/01/2016
                             <1>       ; 29/01/2016
                             <1>       ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 2004-2005
                             <1>
                             <1>       ; DH = cursor row, DL = cursor column
                             <1>       ; BH = 0 = video page number (active page)
                             <1>
                             <1>       ;xor   bh, bh ; 0 = video page 0
                             <1> 
                             <1> readnextchar:
000070F5 30E4                <1>       xor     ah, ah
000070F7 E81A9BFFFF          <1>       call   int16h
000070FC 20C0                <1>       and    al, al
000070FE 7432                <1>       jz     short loc_arrow
00007100 3CE0                <1>       cmp    al, 0E0h
00007102 742E                <1>       je     short loc_arrow
00007104 3C08                <1>       cmp    al, 08h
00007106 7542                <1>       jne    short char_return
                             <1> loc_back:
00007108 3A15[5E590100]      <1>       cmp    dl, [CursorColumn]
0000710E 76E5                <1>       jna     short readnextchar
                             <1> prev_column:
00007110 FECA                <1>       dec    dl
                             <1> set_cursor_pos:
                             <1>       ;push  dx
00007112 52                  <1>       push   edx ; 29/12/2017
                             <1>       ;xor   bh, bh ; 0 = video page 0
                             <1>       ; DH = Row, DL = Column
00007113 E830ACFFFF          <1>       call   _set_cpos ; 17/01/2016
00007118 5A                  <1>         pop edx ; 29/12/2017
                             <1>       ;pop   dx
                             <1>       ;movzx ebx, dl
00007119 88D3                <1>       mov    bl, dl
0000711B 2A1D[5E590100]      <1>       sub    bl, [CursorColumn]
00007121 B020                <1>       mov    al, 20h
00007123 8883[AE590100]      <1>       mov    [CommandBuffer+ebx], al
                             <1>       ;sub   bh, bh ; video page 0
                             <1>       ;mov   cx, 1
00007129 B307                <1>       mov    bl, 7 ; color attribute
0000712B E809ABFFFF          <1>       call   _write_c_current ; 17/01/2016
                             <1>       ;mov   dx, [CURSOR_POSN]
00007130 EBC3                <1>       jmp    short readnextchar
                             <1> loc_arrow:
00007132 80FC4B              <1>       cmp    ah, 4Bh
00007135 74D1                <1>       je     short loc_back
00007137 80FC53              <1>       cmp    ah, 53h
0000713A 74CC                <1>       je      short loc_back
0000713C 80FC4D              <1>       cmp    ah, 4Dh
0000713F 75B4                <1>       jne    short readnextchar
00007141 80FA4F              <1>       cmp    dl, 79
00007144 73AF                <1>       jnb    short readnextchar
00007146 FEC2                <1>       inc    dl
00007148 EBC8                <1>       jmp    short set_cursor_pos
                             <1> char_return:
0000714A 0FB6DA              <1>       movzx  ebx, dl
0000714D 2A1D[5E590100]      <1>       sub    bl, [CursorColumn]
00007153 3C20                <1>       cmp    al, 20h
00007155 721D                <1>       jb     short loc_escape
00007157 8883[AE590100]      <1>       mov    [CommandBuffer+ebx], al
0000715D 80FA4F              <1>       cmp    dl, 79
00007160 7393                <1>       jnb    short readnextchar
00007162 66BB0700            <1>       mov    bx, 7 ; color attribute
00007166 E847ABFFFF          <1>       call   _write_tty
0000716B 668B15[56580100]    <1>       mov    dx, [CURSOR_POSN] ; video page 0
00007172 EB81                <1>         jmp     readnextchar
                             <1> loc_escape:
00007174 3C1B                <1>       cmp    al, 1Bh
00007176 7418                <1>       je     short rw_char_retn
                             <1>       ;
00007178 3C0D                <1>       cmp    al, 0Dh ; CR
0000717A 0F8575FFFFFF        <1>         jne     readnextchar
                             <1>       ; 13/05/2016
00007180 66BB0700            <1>       mov    bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh=0)
00007184 E829ABFFFF          <1>       call   _write_tty
                             <1>       ;mov   bx, 7  ; attribute/color
                             <1>                    ; video page 0 (bh=0)
00007189 B00A                <1>       mov    al, 0Ah ; LF
0000718B E822ABFFFF          <1>       call   _write_tty
                             <1> rw_char_retn:
00007190 C3                  <1>       retn
                             <1> 
                             <1> show_date:
                             <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>         ; 2004-2005
                             <1> 
                             <1>       ;mov   ah, 04h
                             <1>       ;call  int1Ah
00007191 E8C9E8FFFF          <1>       call   RTC_40 ; GET RTC DATE
                             <1> 
00007196 88D0                <1>       mov    al, dl
00007198 E8709AFFFF          <1>       call   bcd_to_ascii
0000719D 66A3[120E0100]      <1>       mov    [Day], ax
                             <1> 
000071A3 88F0                <1>       mov    al, dh
000071A5 E8639AFFFF          <1>       call   bcd_to_ascii
000071AA 66A3[150E0100]      <1>       mov    [Month], ax
                             <1> 
000071B0 88E8                <1>       mov    al, ch
000071B2 E8569AFFFF          <1>       call   bcd_to_ascii
000071B7 66A3[180E0100]      <1>       mov    [Century], ax
                             <1> 
000071BD 88C8                <1>       mov    al, cl
000071BF E8499AFFFF          <1>       call   bcd_to_ascii
000071C4 66A3[1A0E0100]      <1>       mov    word [Year], ax
                             <1> 
000071CA BE[020E0100]        <1>       mov    esi, Msg_Show_Date
000071CF E889F1FFFF          <1>       call   print_msg
                             <1> 
000071D4 C3                  <1>       retn
```

```
                             <1>
                             <1> set_date:
                             <1>       ; 13/05/2016
                             <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>         ; 2004-2005
                             <1>
000071D5 BE[E60D0100]        <1>       mov    esi, Msg_Enter_Date
000071DA E87EF1FFFF          <1>       call   print_msg
                             <1>
                             <1> loc_enter_day_1:
000071DF 30E4                <1>       xor     ah, ah
000071E1 E8309AFFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
000071E6 3C0D                <1>       cmp    al, 13
000071E8 0F84B7010000        <1>       je     loc_set_date_retn
000071EE 3C1B                <1>       cmp    al, 27
000071F0 0F84AF010000        <1>       je     loc_set_date_retn
000071F6 A2[120E0100]        <1>       mov    [Day], al
000071FB 3C30                <1>       cmp    al, '0'
000071FD 0F82AD010000        <1>       jb     loc_set_date_stc_0
00007203 3C33                <1>       cmp    al, '3'
00007205 0F87A5010000        <1>       ja     loc_set_date_stc_0
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                   ; video page 0 (bh)
0000720B B307                <1>       mov    bl, 7
0000720D E8A0AAFFFF          <1>       call   _write_tty
                             <1> loc_enter_day_2:
00007212 30E4                <1>       xor     ah, ah
00007214 E8FD99FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00007219 3C1B                <1>       cmp    al, 27
0000721B 0F8484010000        <1>         je      loc_set_date_retn
00007221 A2[130E0100]        <1>       mov    [Day+1], al
00007226 3C30                <1>       cmp    al, '0'
00007228 0F828C010000        <1>         jb      loc_set_date_stc_1
0000722E 3C39                <1>       cmp    al, '9'
00007230 0F8784010000        <1>         ja      loc_set_date_stc_1
00007236 803D[120E0100]33    <1>       cmp    byte [Day], '3'
0000723D 7208                <1>       jb     short pass_set_day_31
0000723F 3C31                <1>       cmp    al, '1'
00007241 0F8773010000        <1>         ja      loc_set_date_stc_1
                             <1> pass_set_day_31:
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                   ; video page 0 (bh)
00007247 B307                <1>       mov    bl, 7
00007249 E864AAFFFF          <1>       call   _write_tty
                             <1> loc_enter_separator_1:
0000724E 28E4                <1>       sub     ah, ah ; 0
00007250 E8C199FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00007255 3C1B                <1>       cmp    al, 27
00007257 0F8448010000        <1>         je      loc_set_date_retn
0000725D 3C2D                <1>       cmp    al, '-'
0000725F 7408                <1>       je     short pass_set_date_separator_1
00007261 3C2F                <1>       cmp    al, '/'
00007263 0F856C010000        <1>         jne     loc_set_date_stc_2
                             <1> pass_set_date_separator_1:
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                   ; video page 0 (bh)
00007269 B307                <1>       mov    bl, 7
0000726B E842AAFFFF          <1>       call   _write_tty
                             <1> loc_enter_month_1:
00007270 30E4                <1>       xor     ah, ah ; 0
00007272 E89F99FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00007277 3C1B                <1>       cmp    al, 27
00007279 0F8426010000        <1>         je      loc_set_date_retn
0000727F A2[150E0100]        <1>       mov    [Month], al
00007284 3C30                <1>       cmp    al, '0'
00007286 0F8264010000        <1>         jb      loc_set_date_stc_3
0000728C 3C31                <1>       cmp    al, '1'
0000728E 0F875C010000        <1>         ja      loc_set_date_stc_3
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                   ; video page 0 (bh)
00007294 B307                <1>       mov    bl, 7
00007296 E817AAFFFF          <1>       call   _write_tty
                             <1> loc_enter_month_2:
0000729B 30E4                <1>       xor     ah, ah
0000729D E87499FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
000072A2 3C1B                <1>       cmp    al, 27
000072A4 0F84FB000000        <1>         je      loc_set_date_retn
000072AA A2[160E0100]        <1>       mov    [Month+1], al
000072AF 3C30                <1>       cmp    al, '0'
000072B1 0F8254010000        <1>         jb      loc_set_date_stc_4
000072B7 3C39                <1>       cmp    al, '9'
000072B9 0F874C010000        <1>         ja      loc_set_date_stc_4
000072BF 803D[150E0100]31    <1>       cmp    byte [Month], '1'
000072C6 7208                <1>       jb     short pass_set_month_12
000072C8 3C32                <1>       cmp    al, '2'
000072CA 0F873B010000        <1>         ja      loc_set_date_stc_4
                             <1> pass_set_month_12:
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                   ; video page 0 (bh)
000072D0 B307                <1>       mov    bl, 7
000072D2 E8DBA9FFFF          <1>       call   _write_tty
                             <1> loc_enter_separator_2:
000072D7 28E4                <1>       sub     ah, ah
```

```
000072D9 E83899FFFF             <1>       call   int16h
                                <1>       ; AL = ASCII Code of the Character
000072DE 3C1B                   <1>       cmp    al, 27
000072E0 0F84BF000000           <1>         je       loc_set_date_retn
000072E6 3C2D                   <1>       cmp    al, '-'
000072E8 7408                   <1>       je     short pass_set_date_separator_2
000072EA 3C2F                   <1>       cmp    al, '/'
000072EC 0F8534010000           <1>         jne      loc_set_date_stc_5
                                <1> pass_set_date_separator_2:
                                <1>       ; 13/05/2016
                                <1>       ;mov   bx, 7 ; attribute/color (bl)
                                <1>                    ; video page 0 (bh)
000072F2 B307                   <1>       mov    bl, 7
000072F4 E8B9A9FFFF             <1>       call   _write_tty
                                <1> loc_enter_year_1:
000072F9 30E4                   <1>       xor     ah, ah
000072FB E81699FFFF             <1>       call   int16h
                                <1>       ; AL = ASCII Code of the Character
00007300 3C1B                   <1>       cmp    al, 27
00007302 0F849D000000           <1>         je       loc_set_date_retn
00007308 A2[1A0E0100]           <1>       mov    [Year], al
0000730D 3C30                   <1>       cmp    al, '0'
0000730F 0F822C010000           <1>         jb       loc_set_date_stc_6
00007315 3C39                   <1>       cmp    al, '9'
00007317 0F8724010000           <1>         ja       loc_set_date_stc_6
                                <1>       ; 13/05/2016
                                <1>       ;mov   bx, 7 ; attribute/color (bl)
                                <1>                    ; video page 0 (bh)
0000731D B307                   <1>       mov    bl, 7
0000731F E88EA9FFFF             <1>       call   _write_tty
                                <1> loc_enter_year_2:
00007324 30E4                   <1>       xor    ah, ah
00007326 E8EB98FFFF             <1>       call   int16h
                                <1>       ; AL = ASCII Code of the Character
0000732B 3C1B                   <1>       cmp    al, 27
0000732D 7476                   <1>       je     short loc_set_date_retn
0000732F A2[1B0E0100]           <1>       mov    byte [Year+1], al
00007334 3C30                   <1>       cmp    al, '0'
00007336 0F8220010000           <1>         jb       loc_set_date_stc_7
0000733C 3C39                   <1>       cmp    al, '9'
0000733E 0F8718010000           <1>         ja       loc_set_date_stc_7
                                <1>       ; 13/05/2016
                                <1>       ;mov   bx, 7 ; attribute/color (bl)
                                <1>                    ; video page 0 (bh)
00007344 B307                   <1>       mov    bl, 7
00007346 E867A9FFFF             <1>       call   _write_tty
                                <1> loc_set_date_get_lchar_again:
0000734B 28E4                   <1>       sub    ah, ah ; 0
0000734D E8C498FFFF             <1>       call   int16h
                                <1>       ; AL = ASCII Code of the Character
00007352 3C0D                   <1>       cmp    al, 13 ; ENTER key
00007354 7412                   <1>       je     short loc_set_date_progress
00007356 3C1B                   <1>       cmp    al, 27 ; ESC key
00007358 744B                   <1>       je     short loc_set_date_retn
                                <1>       ;
0000735A E82A010000             <1>       call   check_for_backspace
0000735F 75EA                   <1>       jne    short loc_set_date_get_lchar_again
                                <1>
                                <1> loc_set_date_bs_8:
00007361 E811010000             <1>       call   write_backspace
00007366 EBBC                   <1>       jmp    short loc_enter_year_2
                                <1>
                                <1> loc_set_date_progress:
                                <1>       ; Get Current Date
                                <1>       ;mov   ah, 04h
                                <1>       ;call  int1Ah
00007368 E8F2E6FFFF             <1>       call   RTC_40 ; GET RTC DATE
                                <1>       ; CH = century (in BCD)
                                <1>
0000736D 66A1[1A0E0100]         <1>       mov    ax, [Year]
00007373 662D3030               <1>       sub    ax, '00'
00007377 C0E004                 <1>       shl    al, 4 ; * 16
0000737A 88C1                   <1>       mov    cl, al
0000737C 00E1                   <1>       add    cl, ah
0000737E 66A1[150E0100]         <1>       mov    ax, [Month]
00007384 662D3030               <1>       sub    ax, '00'
00007388 C0E004                 <1>       shl    al, 4 ; * 16
0000738B 88C6                   <1>       mov    dh, al
0000738D 00E6                   <1>       add    dh, ah
0000738F 66A1[120E0100]         <1>       mov    ax, [Day]
00007395 662D3030               <1>       sub    ax, '00'
00007399 C0E004                 <1>       shl    al, 4 ; * 16
0000739C 88C2                   <1>       mov    dl, al
0000739E 00E2                   <1>       add    dl, ah
                                <1>
                                <1>       ;mov   ah, 05h
                                <1>       ;call  int1Ah
000073A0 E8E7E6FFFF             <1>       call   RTC_50 ; SET RTC DATE
                                <1>
                                <1> loc_set_date_retn:
000073A5 BE[6F190100]           <1>       mov    esi, nextline
000073AA E8AEEFFFFF             <1>       call   print_msg
000073AF C3                     <1>       retn
                                <1>
                                <1> loc_set_date_stc_0:
                                <1>       ;xor   bh, bh ; video page 0
000073B0 E8DDA9FFFF             <1>       call   beeper ; BEEP !
000073B5 E925FEFFFF             <1>         jmp      loc_enter_day_1
                                <1> loc_set_date_stc_1:
000073BA E8CA000000             <1>       call   check_for_backspace
000073BF 740A                   <1>       je     short loc_set_date_bs_1
                                <1>       ;xor   bh, bh ; video page 0
000073C1 E8CCA9FFFF             <1>       call   beeper ; BEEP !
```

```
000073C6 E947FEFFFF           <1>         jmp     loc_enter_day_2
                              <1> loc_set_date_bs_1:
000073CB E8A7000000           <1>       call   write_backspace
000073D0 E90AFEFFFF           <1>         jmp     loc_enter_day_1
                              <1> loc_set_date_stc_2:
000073D5 E8AF000000           <1>       call   check_for_backspace
000073DA 740A                 <1>       je     short loc_set_date_bs_2
                              <1>       ;xor   bh, bh ; video page 0
000073DC E8B1A9FFFF           <1>       call   beeper ; BEEP !
000073E1 E968FEFFFF           <1>         jmp     loc_enter_separator_1
                              <1> loc_set_date_bs_2:
000073E6 E88C000000           <1>       call   write_backspace
000073EB E922FEFFFF           <1>         jmp     loc_enter_day_2
                              <1> loc_set_date_stc_3:
000073F0 E894000000           <1>       call   check_for_backspace
000073F5 740A                 <1>       je short loc_set_date_bs_3
                              <1>       ;xor   bh, bh ; video page 0
000073F7 E896A9FFFF           <1>       call   beeper ; BEEP !
000073FC E96FFEFFFF           <1>         jmp     loc_enter_month_1
                              <1> loc_set_date_bs_3:
00007401 E871000000           <1>       call   write_backspace
00007406 E943FEFFFF           <1>         jmp     loc_enter_separator_1
                              <1> loc_set_date_stc_4:
0000740B E879000000           <1>       call   check_for_backspace
00007410 740A                 <1>       je     short loc_set_date_bs_4
                              <1>       ;xor   bh, bh ; video page 0
00007412 E87BA9FFFF           <1>       call   beeper ; BEEP !
00007417 E97FFEFFFF           <1>         jmp     loc_enter_month_2
                              <1> loc_set_date_bs_4:
0000741C E856000000           <1>       call   write_backspace
00007421 E94AFEFFFF           <1>         jmp     loc_enter_month_1
                              <1> loc_set_date_stc_5:
00007426 E85E000000           <1>       call   check_for_backspace
0000742B 740A                 <1>       je     short loc_set_date_bs_5
                              <1>       ;xor   bh, bh ; video page 0
0000742D E860A9FFFF           <1>       call   beeper ; BEEP !
00007432 E9A0FEFFFF           <1>         jmp     loc_enter_separator_2
                              <1> loc_set_date_bs_5:
00007437 E83B000000           <1>       call   write_backspace
0000743C E95AFEFFFF           <1>         jmp     loc_enter_month_2
                              <1> loc_set_date_stc_6:
00007441 E843000000           <1>       call   check_for_backspace
00007446 740A                 <1>         je      short   loc_set_date_bs_6
                              <1>       ;xor   bh, bh ; video page 0
00007448 E845A9FFFF           <1>       call   beeper ; BEEP !
0000744D E9A7FEFFFF           <1>         jmp     loc_enter_year_1
                              <1> loc_set_date_bs_6:
00007452 E820000000           <1>       call   write_backspace
00007457 E97BFEFFFF           <1>         jmp     loc_enter_separator_2
                              <1> loc_set_date_stc_7:
0000745C E828000000           <1>       call   check_for_backspace
00007461 740A                 <1>       je     short loc_set_date_bs_7
                              <1>       ;xor   bh, bh ; video page 0
00007463 E82AA9FFFF           <1>       call   beeper ; BEEP !
00007468 E9B7FEFFFF           <1>         jmp     loc_enter_year_2
                              <1> loc_set_date_bs_7:
0000746D E805000000           <1>       call   write_backspace
00007472 E982FEFFFF           <1>         jmp     loc_enter_year_1
                              <1>
                              <1> write_backspace:
                              <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
00007477 B008                 <1>       mov    al, 08h ; BACKSPACE
                              <1>       ; 13/05/2016
00007479 66BB0700             <1>       mov    bx, 7 ; bl = attribute/color
                              <1>                   ; bh = video page = 0
0000747D E830A8FFFF           <1>       call   _write_tty
00007482 B020                 <1>       mov    al, 20h ; BLANK/SPACE char
                              <1>       ;mov   bx, 7 ; attribute/color
                              <1>       ;call _write_c_current
                              <1>       ;retn
00007484 E9B0A7FFFF           <1>       jmp    _write_c_current
                              <1>
                              <1> check_for_backspace:
                              <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
00007489 663D080E             <1>       cmp    ax, 0E08h
0000748D 7410                 <1>       je     short cfbs_retn
0000748F 663DE04B             <1>       cmp    ax, 4BE0h
00007493 740A                 <1>       je     short cfbs_retn
00007495 663D004B             <1>       cmp    ax, 4B00h
00007499 7404                 <1>       je     short cfbs_retn
0000749B 663DE053             <1>       cmp    ax, 53E0h
                              <1> cfbs_retn:
0000749F C3                   <1>       retn
                              <1>
                              <1> show_time:
                              <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
                              <1>         ; 2004-2005
                              <1>
                              <1>       ;mov   ah, 02h
                              <1>       ;call  int1Ah
000074A0 E849E5FFFF           <1>       call   RTC_20 ; GET RTC TIME
                              <1>
000074A5 88E8                 <1>       mov    al, ch
000074A7 E86197FFFF           <1>       call   bcd_to_ascii
000074AC 66A3[400E0100]       <1>       mov    [Hour], ax
                              <1>
000074B2 88C8                 <1>       mov    al, cl
000074B4 E85497FFFF           <1>       call   bcd_to_ascii
000074B9 66A3[430E0100]       <1>       mov    [Minute], ax
                              <1>
000074BF 88F0                 <1>       mov    al, dh
000074C1 E84797FFFF           <1>       call   bcd_to_ascii
000074C6 66A3[460E0100]       <1>       mov    [Second], ax
```

```
                             <1>
000074CC BE[300E0100]        <1>       mov    esi, Msg_Show_Time
000074D1 E887EEFFFF          <1>       call   print_msg
000074D6 C3                  <1>       retn
                             <1>
                             <1> set_time:
                             <1>       ; 13/05/2016
                             <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>         ; 2004-2005
                             <1>
000074D7 BE[1F0E0100]        <1>       mov    esi, Msg_Enter_Time
000074DC E87CEEFFFF          <1>       call   print_msg
                             <1>
                             <1> loc_enter_hour_1:
000074E1 30E4                <1>       xor     ah, ah
000074E3 E82E97FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
000074E8 3C0D                <1>       cmp    al, 13 ; ENTER key
000074EA 0F84AE010000        <1>         je      loc_set_time_retn
000074F0 3C1B                <1>       cmp    al, 27 ; ESC key
000074F2 0F84A6010000        <1>         je      loc_set_time_retn
000074F8 A2[400E0100]        <1>       mov    [Hour], al
000074FD 3C30                <1>       cmp    al, '0'
000074FF 0F82A4010000        <1>         jb      loc_set_time_stc_0
00007505 3C32                <1>       cmp    al, '2'
00007507 0F879C010000        <1>         ja      loc_set_time_stc_0
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
0000750D B307                <1>       mov    bl, 7
0000750F E89EA7FFFF          <1>       call   _write_tty
                             <1> loc_enter_hour_2:
00007514 30E4                <1>       xor     ah, ah
00007516 E8FB96FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
0000751B 3C1B                <1>       cmp    al, 27
0000751D 0F847B010000        <1>         je      loc_set_time_retn
00007523 A2[410E0100]        <1>       mov    [Hour+1], al
00007528 3C30                <1>       cmp    al, '0'
0000752A 0F8283010000        <1>         jb      loc_set_time_stc_1
00007530 3C39                <1>       cmp    al, '9'
00007532 0F877B010000        <1>       ja     loc_set_time_stc_1
00007538 803D[400E0100]32    <1>         cmp     byte [Hour], '2'
0000753F 7208                <1>       jb     short pass_set_time_24
00007541 3C34                <1>       cmp    al, '4'
00007543 0F876A010000        <1>         ja      loc_set_time_stc_1
                             <1> pass_set_time_24:
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
00007549 B307                <1>       mov    bl, 7
0000754B E862A7FFFF          <1>       call   _write_tty
                             <1> loc_enter_time_separator_1:
00007550 28E4                <1>       sub    ah, ah ; 0
00007552 E8BF96FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00007557 3C1B                <1>       cmp    al, 27
00007559 0F843F010000        <1>         je      loc_set_time_retn
0000755F 3C3A                <1>       cmp    al, ':'
00007561 0F8567010000        <1>         jne     loc_set_time_stc_2
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
00007567 B307                <1>       mov    bl, 7
00007569 E844A7FFFF          <1>       call   _write_tty
                             <1> loc_enter_minute_1:
0000756E 30E4                <1>       xor     ah, ah
00007570 E8A196FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00007575 3C1B                <1>       cmp    al, 27
00007577 0F8421010000        <1>         je      loc_set_time_retn
0000757D A2[430E0100]        <1>       mov    [Minute], al
00007582 3C30                <1>       cmp    al, '0'
00007584 0F825F010000        <1>         jb      loc_set_time_stc_3
0000758A 3C35                <1>       cmp    al, '5'
0000758C 0F8757010000        <1>         ja      loc_set_time_stc_3
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
00007592 B307                <1>       mov    bl, 7
00007594 E819A7FFFF          <1>       call   _write_tty
                             <1> loc_enter_minute_2:
00007599 30E4                <1>       xor     ah, ah
0000759B E87696FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
000075A0 3C1B                <1>       cmp    al, 27
000075A2 0F84F6000000        <1>         je      loc_set_time_retn
000075A8 A2[440E0100]        <1>       mov    [Minute+1], al
000075AD 3C30                <1>       cmp    al, '0'
000075AF 0F824F010000        <1>         jb      loc_set_time_stc_4
000075B5 3C39                <1>       cmp    al, '9'
000075B7 0F8747010000        <1>         ja      loc_set_time_stc_4
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
000075BD B307                <1>       mov    bl, 7
000075BF E8EEA6FFFF          <1>       call   _write_tty
                             <1> loc_enter_time_separator_2:
000075C4 66C705[460E0100]30- <1>       mov    word [Second], 3030h
000075CC 30                  <1>
000075CD 28E4                <1>       sub     ah, ah
000075CF E84296FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
```

```
000075D4 3C0D                <1>       cmp    al, 13
000075D6 0F8485000000        <1>         je      loc_set_time_progress
000075DC 3C1B                <1>       cmp    al, 27
000075DE 0F84BA000000        <1>         je      loc_set_time_retn
000075E4 3C3A                <1>       cmp    al, ':'
000075E6 0F8533010000        <1>         jne     loc_set_time_stc_5
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
000075EC B307                <1>       mov    bl, 7
000075EE E8BFA6FFFF          <1>       call   _write_tty
                             <1> loc_enter_second_1:
000075F3 30E4                <1>       xor     ah, ah
000075F5 E81C96FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
000075FA 3C0D                <1>       cmp    al, 13
000075FC 7463                <1>       je     short loc_set_time_progress
000075FE 3C1B                <1>       cmp    al, 27
00007600 0F8498000000        <1>         je      loc_set_time_retn
00007606 A2[460E0100]        <1>       mov    [Second], al
0000760B 3C30                <1>       cmp    al, '0'
0000760D 0F8227010000        <1>         jb      loc_set_time_stc_6
00007613 3C35                <1>       cmp    al, '5'
00007615 0F871F010000        <1>         ja      loc_set_time_stc_6
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
0000761B B307                <1>       mov    bl, 7
0000761D E890A6FFFF          <1>       call   _write_tty
                             <1> loc_enter_second_2:
00007622 30E4                <1>       xor     ah, ah
00007624 E8ED95FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
00007629 3C1B                <1>       cmp    al, 27
0000762B 7471                <1>       je     short loc_set_time_retn
0000762D 3C30                <1>       cmp    al, '0'
0000762F 0F8229010000        <1>         jb      loc_set_time_stc_7
00007635 3C39                <1>       cmp    al, '9'
00007637 0F8721010000        <1>         ja      loc_set_time_stc_7
                             <1>       ; 13/05/2016
                             <1>       ;mov   bx, 7 ; attribute/color (bl)
                             <1>                    ; video page 0 (bh)
0000763D B307                <1>       mov    bl, 7
0000763F E86EA6FFFF          <1>       call   _write_tty
                             <1> loc_set_time_get_lchar_again:
00007644 28E4                <1>       sub    ah, ah ; 0
00007646 E8CB95FFFF          <1>       call   int16h
                             <1>       ; AL = ASCII Code of the Character
0000764B 3C0D                <1>       cmp    al, 13
0000764D 7412                <1>       je     short loc_set_time_progress
0000764F 3C1B                <1>       cmp    al, 27
00007651 744B                <1>       je     short loc_set_time_retn
                             <1>       ;
00007653 E831FEFFFF          <1>       call   check_for_backspace
00007658 75EA                <1>       jne    short loc_set_time_get_lchar_again
                             <1>
                             <1> loc_set_time_bs_8:
0000765A E818FEFFFF          <1>       call   write_backspace
0000765F EBC1                <1>       jmp    short loc_enter_second_2
                             <1>
                             <1> loc_set_time_progress:
                             <1>       ; Get Current Time
                             <1>       ;mov   ah, 02h
                             <1>       ;call  int1Ah
00007661 E888E3FFFF          <1>       call   RTC_20 ; GET RTC TIME
                             <1>       ;DL = Daylight Savings Enable option (0-1)
                             <1>
00007666 66A1[400E0100]      <1>       mov    ax, [Hour]
0000766C 662D3030            <1>       sub    ax, '00'
00007670 C0E004              <1>       shl    al, 4 ; * 16
00007673 88C5                <1>       mov    ch, al
00007675 00E5                <1>       add    ch, ah
00007677 66A1[430E0100]      <1>       mov    ax, [Minute]
0000767D 662D3030            <1>       sub    ax, '00'
00007681 C0E004              <1>       shl    al, 4 ; * 16
00007684 88C1                <1>       mov    cl, al
00007686 00E1                <1>       add    cl, ah
00007688 66A1[460E0100]      <1>       mov    ax, [Second]
0000768E 662D3030            <1>       sub    ax, '00'
00007692 C0E004              <1>       shl    al, 4 ; * 16
00007695 88C6                <1>       mov    dh, al
00007697 00E6                <1>       add    dh, ah
                             <1>
                             <1>       ;mov   ah, 03h
                             <1>       ;call  int1Ah
00007699 E87FE3FFFF          <1>       call   RTC_30 ; SET RTC TIME
                             <1>
                             <1> loc_set_time_retn:
0000769E BE[6F190100]        <1>       mov    esi, nextline
000076A3 E8B5ECFFFF          <1>       call   print_msg
000076A8 C3                  <1>       retn
                             <1>
                             <1> loc_set_time_stc_0:
                             <1>       ;xor   bh, bh ; video page 0
000076A9 E8E4A6FFFF          <1>       call   beeper ; BEEP !
000076AE E92EFEFFFF          <1>         jmp     loc_enter_hour_1
                             <1> loc_set_time_stc_1:
000076B3 E8D1FDFFFF          <1>       call   check_for_backspace
000076B8 740A                <1>       je     short loc_set_time_bs_1
                             <1>       ;xor   bh, bh ; video page 0
000076BA E8D3A6FFFF          <1>       call   beeper ; BEEP !
000076BF E950FEFFFF          <1>         jmp     loc_enter_hour_2
                             <1> loc_set_time_bs_1:
```

```
000076C4 E8AEFDFFFF          <1>       call   write_backspace
000076C9 E913FEFFFF          <1>         jmp     loc_enter_hour_1
                             <1> loc_set_time_stc_2:
000076CE E8B6FDFFFF          <1>       call   check_for_backspace
000076D3 740A                <1>       je     short loc_set_time_bs_2
                             <1>       ;xor    bh, bh ; video page 0
000076D5 E8B8A6FFFF          <1>       call   beeper ; BEEP !
000076DA E971FEFFFF          <1>         jmp     loc_enter_time_separator_1
                             <1> loc_set_time_bs_2:
000076DF E893FDFFFF          <1>       call   write_backspace
000076E4 E92BFEFFFF          <1>         jmp     loc_enter_hour_2
                             <1> loc_set_time_stc_3:
000076E9 E89BFDFFFF          <1>       call   check_for_backspace
000076EE 740A                <1>       je     short loc_set_time_bs_3
                             <1>       ;xor    bh, bh ; video page 0
000076F0 E89DA6FFFF          <1>       call   beeper ; BEEP !6
000076F5 E974FEFFFF          <1>         jmp     loc_enter_minute_1
                             <1> loc_set_time_bs_3:
000076FA E878FDFFFF          <1>       call   write_backspace
000076FF E94CFEFFFF          <1>         jmp     loc_enter_time_separator_1
                             <1> loc_set_time_stc_4:
00007704 E880FDFFFF          <1>       call   check_for_backspace
00007709 740A                <1>       je     short loc_set_time_bs_4
                             <1>       ;xor    bh, bh ; video page 0
0000770B E882A6FFFF          <1>       call   beeper ; BEEP !
00007710 E984FEFFFF          <1>         jmp     loc_enter_minute_2
                             <1> loc_set_time_bs_4:
00007715 E85DFDFFFF          <1>       call   write_backspace
0000771A E94FFEFFFF          <1>         jmp     loc_enter_minute_1
                             <1> loc_set_time_stc_5:
0000771F E865FDFFFF          <1>       call   check_for_backspace
00007724 740A                <1>       je     short loc_set_time_bs_5
                             <1>       ;xor    bh, bh ; video page 0
00007726 E867A6FFFF          <1>       call   beeper ; BEEP !
0000772B E994FEFFFF          <1>         jmp     loc_enter_time_separator_2
                             <1> loc_set_time_bs_5:
00007730 E842FDFFFF          <1>       call   write_backspace
00007735 E95FFEFFFF          <1>         jmp     loc_enter_minute_2
                             <1> loc_set_time_stc_6:
0000773A E84AFDFFFF          <1>       call   check_for_backspace
0000773F 7413                <1>       je     short loc_set_time_bs_6
                             <1>       ;xor    bh, bh ; video page 0
00007741 E84CA6FFFF          <1>       call   beeper ; BEEP !
00007746 66C705[460E0100]30- <1>       mov    word [Second], 3030h
0000774E 30                  <1>
0000774F E99FFEFFFF          <1>         jmp     loc_enter_second_1
                             <1> loc_set_time_bs_6:
00007754 E81EFDFFFF          <1>       call   write_backspace
00007759 E966FEFFFF          <1>         jmp     loc_enter_time_separator_2
                             <1> loc_set_time_stc_7:
0000775E E826FDFFFF          <1>       call   check_for_backspace
00007763 740A                <1>       je     short loc_set_time_bs_7
                             <1>       ;xor    bh, bh ; video page 0
00007765 E828A6FFFF          <1>       call   beeper ; BEEP !
0000776A E9B3FEFFFF          <1>         jmp     loc_enter_second_2
                             <1> loc_set_time_bs_7:
0000776F E803FDFFFF          <1>       call   write_backspace
00007774 E97AFEFFFF          <1>         jmp     loc_enter_second_1
                             <1>
                             <1> print_volume_info:
                             <1>       ; 01/03/2016
                             <1>       ; 08/02/2016
                             <1>       ; 06/02/2016
                             <1>       ; 04/02/2016
                             <1>       ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 25/10/2009
                             <1>       ;
                             <1>       ; "Volume Serial No: "
                             <1>       ;
                             <1>       ; INPUT  : AL = DOS Drive Number
                             <1>       ; OUTPUT : AH = FS Type
                             <1>       ;          AL = DOS Drive Name
                             <1>       ; CF = 0 -> OK
                             <1>       ; CF = 1 -> Drive not ready
                             <1>
00007779 88C4                <1>       mov    ah, al
0000777B 28C0                <1>       sub    al, al
0000777D 0FB7F0              <1>       movzx esi, ax
00007780 81C600010900        <1>       add    esi, Logical_DOSDisks
00007786 8A06                <1>       mov    al, [esi]
00007788 3C41                <1>       cmp    al, 'A'
0000778A 7304                <1>       jnb    short loc_pvi_set_vol_name
0000778C 8A6604              <1>       mov    ah, [esi+LD_FSType]
0000778F C3                  <1>       retn
                             <1>
                             <1> loc_pvi_set_vol_name:
00007790 A2[7A0E0100]        <1>       mov    [Vol_Drv_Name], al
00007795 56                  <1>       push   esi
00007796 E858010000          <1>       call   move_volume_name_and_serial_no ;;;
0000779B 7302                <1>       jnc    short loc_pvi_mvn_ok
0000779D 5E                  <1>       pop    esi
0000779E C3                  <1>       retn
                             <1>
                             <1> loc_pvi_mvn_ok:
0000779F 8B3424              <1>       mov    esi, [esp]
000077A2 807E04A1            <1>       cmp    byte [esi+LD_FSType], 0A1h
000077A6 7509                <1>       jne    short loc_pvi_fat_vol_size
000077A8 8B4670              <1>       mov    eax, [esi+LD_FS_VolumeSize]
000077AB 0FB75E11            <1>       movzx ebx, word [esi+LD_FS_BytesPerSec]
000077AF EB07                <1>       jmp    short loc_vol_size_mul32
                             <1> loc_pvi_fat_vol_size:
000077B1 8B4670              <1>       mov    eax, [esi+LD_TotalSectors]
000077B4 0FB75E11            <1>       movzx ebx, word [esi+LD_BPB+BPB_BytsPerSec]
```

```
                             <1> loc_vol_size_mul32:
000077B8 F7E3                <1>       mul    ebx
000077BA 09D2                <1>       or     edx, edx
000077BC 7507                <1>       jnz    short loc_vol_size_in_kbytes
                             <1> loc_vol_size_in_bytes:
000077BE B9[580E0100]        <1>       mov    ecx, VolSize_Bytes
000077C3 EB0D                <1>       jmp    short loc_write_vol_size_str
                             <1> loc_vol_size_in_kbytes:
000077C5 66BB0004            <1>       mov    bx, 1024
000077C9 F7F3                <1>       div    ebx
000077CB B9[4B0E0100]        <1>       mov    ecx, VolSize_KiloBytes
000077D0 31D2                <1>       xor    edx, edx ; 0
                             <1> loc_write_vol_size_str:
000077D2 890D[37610100]      <1>       mov    [VolSize_Unit1], ecx
                             <1>       ;
000077D8 BF[4D610100]        <1>       mov    edi, Vol_Tot_Sec_Str_End
                             <1>         ;mov byte [edi], 0
000077DD B90A000000          <1>       mov    ecx, 10
                             <1> loc_write_vol_size_chr:
000077E2 F7F1                <1>       div    ecx
000077E4 80C230              <1>       add    dl, '0'
000077E7 4F                  <1>       dec    edi
000077E8 8817                <1>       mov    [edi], dl
000077EA 85C0                <1>       test   eax, eax
000077EC 7404                <1>       jz     short loc_write_vol_size_str_ok
000077EE 28D2                <1>       sub    dl, dl ; 0
000077F0 EBF0                <1>       jmp    short loc_write_vol_size_chr
                             <1>
                             <1> loc_write_vol_size_str_ok:
000077F2 893D[3F610100]      <1>       mov    [Vol_Tot_Sec_Str_Start], edi
                             <1>       ;
000077F8 BF[630E0100]        <1>       mov    edi, Vol_FS_Name
000077FD 8A4E03              <1>       mov    cl, [esi+LD_FATType]
00007800 20C9                <1>       and    cl, cl ; 0 ?
00007802 7515                <1>       jnz    short loc_write_vol_FAT_str_1
00007804 66C7075452          <1>       mov    word [edi], 'TR'
00007809 C7470420465331      <1>       mov    dword [edi+4], ' FS1'
                             <1>       ;movzx ebx, word [esi+LD_FS_BytesPerSec]
00007810 668B5E11            <1>       mov    bx, [esi+LD_FS_BytesPerSec]
00007814 8B4674              <1>       mov    eax, [esi+LD_FS_FreeSectors]
00007817 EB36                <1>       jmp    short loc_vol_freespace_mul32
                             <1>
                             <1> loc_write_vol_FAT_str_1:
00007819 66B83332            <1>       mov    ax, '32' ; FAT32
0000781D 80F902              <1>       cmp    cl, 2 ; [esi+LD_FATType]
00007820 7708                <1>       ja     short loc_write_vol_FAT_str_2
00007822 66B83132            <1>       mov    ax, '12' ; FAT12
00007826 7202                <1>       jb     short loc_write_vol_FAT_str_2
00007828 B436                <1>       mov    ah, '6'  ; FAT16
                             <1> loc_write_vol_FAT_str_2:
0000782A C70746415420        <1>       mov    dword [edi], 'FAT '
00007830 66894704            <1>       mov    word [edi+4], ax
                             <1>       ;
                             <1>       ;movzx ebx, word [esi+LD_BPB+BPB_BytsPerSec]
00007834 668B5E11            <1>       mov    bx, [esi+LD_BPB+BPB_BytsPerSec]
00007838 8B4674              <1>       mov    eax, [esi+LD_FreeSectors]
                             <1>
                             <1> loc_vol_freespace_recalc0:
                             <1>       ; 01/03/2016
0000783B 83F8FF              <1>       cmp    eax, 0FFFFFFFFh
0000783E 720F                <1>       jb     short loc_vol_freespace_mul32
                             <1>       ;inc   eax ; 0
00007840 20C9                <1>       and    cl, cl ; byte [esi+LD_FATType]
00007842 740B                <1>       jz     short loc_vol_freespace_mul32
00007844 53                  <1>       push   ebx
00007845 66BB00FF            <1>       mov    bx, 0FF00h ; recalculate free sectors
00007849 E876490000          <1>       call   calculate_fat_freespace
0000784E 5B                  <1>       pop    ebx
                             <1>
                             <1> loc_vol_freespace_mul32:
0000784F F7E3                <1>       mul    ebx
00007851 09D2                <1>       or     edx, edx
00007853 7507                <1>       jnz    short loc_vol_fspace_in_kbytes
                             <1> loc_vol_fspace_in_bytes:
00007855 B9[580E0100]        <1>       mov    ecx, VolSize_Bytes
0000785A EB0D                <1>       jmp    short loc_write_vol_fspace_str
                             <1> loc_vol_fspace_in_kbytes:
0000785C 66BB0004            <1>       mov    bx, 1024
00007860 F7F3                <1>       div    ebx
00007862 B9[4B0E0100]        <1>       mov    ecx, VolSize_KiloBytes
00007867 31D2                <1>       xor    edx, edx ; 0
                             <1> loc_write_vol_fspace_str:
00007869 890D[3B610100]      <1>       mov    [VolSize_Unit2], ecx
                             <1>       ;
0000786F BF[5D610100]        <1>       mov    edi, Vol_Free_Sectors_Str_End
                             <1>         ;mov byte [edi], 0
00007874 B90A000000          <1>       mov    ecx, 10
                             <1> loc_write_vol_fspace_chr:
00007879 F7F1                <1>       div    ecx
0000787B 80C230              <1>       add    dl, '0'
0000787E 4F                  <1>       dec    edi
0000787F 8817                <1>       mov    [edi], dl
00007881 85C0                <1>       test   eax, eax
00007883 7404                <1>       jz     short loc_write_vol_fspace_str_ok
00007885 28D2                <1>       sub    dl, dl ; 0
00007887 EBF0                <1>       jmp    short loc_write_vol_fspace_chr
                             <1>
                             <1> loc_write_vol_fspace_str_ok:
00007889 893D[4F610100]      <1>       mov    [Vol_Free_Sectors_Str_Start], edi
                             <1>       ;
0000788F BE[610E0100]        <1>       mov    esi, Volume_in_drive
00007894 E8C4EAFFFF          <1>       call   print_msg
00007899 BE[A10E0100]        <1>       mov    esi, Vol_Name
```

```
0000789E E8BAEAFFFF          <1>       call   print_msg
000078A3 BE[6F190100]        <1>       mov    esi, nextline
000078A8 E8B0EAFFFF          <1>       call   print_msg
                             <1>       ;
000078AD BE[020F0100]        <1>       mov    esi, Vol_Total_Sector_Header
000078B2 E8A6EAFFFF          <1>       call   print_msg
000078B7 8B35[3F610100]      <1>       mov    esi, [Vol_Tot_Sec_Str_Start]
000078BD E89BEAFFFF          <1>       call   print_msg
000078C2 8B35[37610100]      <1>       mov    esi, [VolSize_Unit1]
000078C8 E890EAFFFF          <1>       call   print_msg
                             <1>       ;
000078CD BE[130F0100]        <1>       mov    esi, Vol_Free_Sectors_Header
000078D2 E886EAFFFF          <1>       call   print_msg
000078D7 8B35[4F610100]      <1>       mov    esi, [Vol_Free_Sectors_Str_Start]
000078DD E87BEAFFFF          <1>       call   print_msg
000078E2 8B35[3B610100]      <1>       mov    esi, [VolSize_Unit2]
000078E8 E870EAFFFF          <1>       call   print_msg
                             <1>       ;
000078ED 5E                  <1>       pop    esi
                             <1>
                             <1>       ;mov   ah, [esi+LD_FSType]
                             <1>       ;mov   al, [esi+LD_FATType]
000078EE 668B4603            <1>       mov    ax, [esi+LD_FATType]
                             <1>
000078F2 C3                  <1>       retn
                             <1>
                             <1> move_volume_name_and_serial_no:
                             <1>       ; 08/02/2016  (TRDOS 386 = TRDOS v2.0)
                             <1>       ; this routine will be called by
                             <1>       ; "print_volume_info" and "print_directory"
                             <1>       ; INPUT ->
                             <1>       ;      ESI = Logical DOS drv descripton table address
                             <1>       ; OUTPUT ->
                             <1>       ;      *Volume name will be moved to text area
                             <1>       ;      *Volume serial number will be converted to
                             <1>       ;       text and will be moved to text area
                             <1>       ;   cf = 1 -> invalid/unknown dos drive
                             <1>       ;   cf = 0 -> ecx = 0
                             <1>       ;
                             <1>       ; (eax, edx, ecx, esi, edi will be changed)
                             <1>
000078F3 BF[A10E0100]        <1>       mov    edi, Vol_Name
                             <1>
                             <1>       ;mov   ah, [esi+LD_FSType]
                             <1>       ;mov   al, [esi+LD_FATType]
000078F8 668B4603            <1>       mov    ax, [esi+LD_FATType]
000078FC 80FCA1              <1>       cmp    ah, 0A1h
000078FF 7418                <1>       je     short mvn_2
00007901 08E4                <1>       or     ah, ah
00007903 7404                <1>       jz     short mvn_0
00007905 08C0                <1>       or     al, al
00007907 7504                <1>       jnz    short mvn_1
                             <1> mvn_0:
00007909 8A06                <1>       mov    al, [esi]
0000790B F9                  <1>       stc
0000790C C3                  <1>       retn
                             <1> mvn_1:
0000790D 3C02                <1>       cmp    al, 2
0000790F 7717                <1>       ja     short mvn_3
                             <1>       ;or    al, al
                             <1>       ;jz    short mvn_2
00007911 8B462D              <1>       mov    eax, [esi+LD_BPB+VolumeID]
00007914 83C631              <1>       add    esi, LD_BPB+VolumeLabel
00007917 EB15                <1>       jmp    short mvn_4
                             <1> mvn_2:
00007919 8B4628              <1>       mov    eax, [esi+LD_FS_VolumeSerial]
0000791C 83C62C              <1>       add    esi, LD_FS_VolumeName
0000791F B910000000          <1>       mov    ecx, 16
00007924 F3A5                <1>       rep    movsd
00007926 EB10                <1>       jmp    short mvn_5
                             <1> mvn_3:
00007928 8B4649              <1>       mov    eax, [esi+LD_BPB+FAT32_VolID]
0000792B 83C64D              <1>       add    esi, LD_BPB+FAT32_VolLab
                             <1> mvn_4:
0000792E B90B000000          <1>       mov    ecx, 11
00007933 F3A4                <1>       rep    movsb
00007935 C60700              <1>       mov    byte [edi], 0
                             <1> mvn_5:
                             <1>       ;mov   [Current_VolSerial], eax
00007938 E8CCB9FFFF          <1>       call   dwordtohex
0000793D 8915[F60E0100]      <1>       mov    [Vol_Serial1], edx
00007943 A3[FB0E0100]        <1>       mov    [Vol_Serial2], eax
                             <1>       ; ecx = 0
00007948 C3                  <1>       retn
                             <1>
                             <1> get_volume_serial_number:
                             <1>       ; 19/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 08/08/2010
                             <1>       ;
                             <1>       ; INPUT -> DL = Logical DOS Drive number
                             <1>       ; OUTPUT -> EAX = Volume serial number
                             <1>       ;           BL= FAT Type
                             <1>       ;           BH = Logical DOS drv Number (DL input)
                             <1>       ; cf = 1 -> Drive not ready
                             <1>
00007949 31DB                <1>       xor    ebx, ebx
0000794B 88D7                <1>       mov    bh, dl
0000794D 3815[D20C0100]      <1>       cmp    [Last_DOS_DiskNo], dl
00007953 7304                <1>       jnb    short loc_gvsn_start
                             <1> loc_gvsn_stc_retn:
00007955 31C0                <1>       xor    eax, eax
00007957 F9                  <1>       stc
00007958 C3                  <1>         retn
```

```
                             <1> loc_gvsn_start:
00007959 56                  <1>       push   esi
0000795A BE00010900          <1>       mov    esi, Logical_DOSDisks
0000795F 01DE                <1>       add    esi, ebx
00007961 8A5E03              <1>       mov    bl, [esi+LD_FATType]
00007964 20DB                <1>       and    bl, bl
00007966 740F                <1>       jz     short loc_gvsn_fs
00007968 80FB02              <1>       cmp    bl, 2
0000796B 7705                <1>       ja     short loc_gvsn_fat32
                             <1> loc_gvsn_fat:
0000796D 83C62D              <1>       add    esi, LD_BPB + VolumeID
00007970 EB0E                <1>       jmp    short loc_gvsn_return
                             <1> loc_gvsn_fat32:
00007972 83C649              <1>       add    esi, LD_BPB + FAT32_VolID
00007975 EB09                <1>       jmp    short loc_gvsn_return
                             <1> loc_gvsn_fs:
00007977 807E04A1            <1>       cmp    byte [esi+LD_FSType], 0A1h
0000797B 75D8                <1>       jne    short loc_gvsn_stc_retn
0000797D 83C628              <1>       add    esi, LD_FS_VolumeSerial
                             <1> loc_gvsn_return:
00007980 8B06                <1>       mov    eax, [esi]
00007982 5E                  <1>       pop    esi
00007983 C3                  <1>       retn
                             <1>
                             <1> ; CMD_INTR.ASM [ TRDOS Command Interpreter Procedure ]
                             <1> ; 09/11/2011
                             <1> ; 29/01/2005
                             <1>
                             <1> command_interpreter:
                             <1>       ; 16/10/2016
                             <1>       ; 12/10/2016
                             <1>       ; 13/05/2016
                             <1>       ; 07/05/2016
                             <1>       ; 04/03/2016
                             <1>       ; 04/02/2016
                             <1>       ; 03/02/2016
                             <1>       ; 30/01/2016
                             <1>       ; 29/01/2016 (TRDOS 386 = TRDOS 2.0)
                             <1>       ; 15/09/2011
                             <1>       ; 29/01/2005
                             <1>
                             <1>       ; Input: ecx = command word length (CL)
                             <1>       ;       CommandBuffer = Command string offset
                             <1>
00007984 C605[F0610100]00    <1>       mov    byte [Program_Exit],0
0000798B 80F904              <1>       cmp    cl, 4
0000798E 0F87B5020000        <1>         ja      c_6
00007994 0F8237010000        <1>         jb      c_2
                             <1> c_4:
                             <1>
                             <1> cmp_cmd_exit:
0000799A BF[400D0100]        <1>       mov    edi, Cmd_Exit
0000799F E8C2030000          <1>       call   cmp_cmd
000079A4 7208                <1>       jc     short cmp_cmd_date
                             <1>
000079A6 C605[F0610100]01    <1>         mov     byte [Program_Exit], 1
000079AD C3                  <1>         retn
                             <1>
                             <1> cmp_cmd_date:
000079AE B104                <1>       mov    cl, 4
000079B0 BF[5C0D0100]        <1>       mov    edi, Cmd_Date
000079B5 E8AC030000          <1>       call   cmp_cmd
000079BA 720B                <1>         jc   short cmp_cmd_time
                             <1>
000079BC E8D0F7FFFF          <1>       call   show_date
000079C1 E80FF8FFFF          <1>       call   set_date
000079C6 C3                  <1>       retn
                             <1>
                             <1> cmp_cmd_time:
000079C7 B104                <1>       mov    cl, 4
000079C9 BF[610D0100]        <1>       mov    edi, Cmd_Time
000079CE E893030000          <1>       call   cmp_cmd
000079D3 720B                <1>       jc     short cmp_cmd_show
                             <1>
000079D5 E8C6FAFFFF          <1>       call   show_time
000079DA E8F8FAFFFF          <1>       call   set_time
000079DF C3                  <1>       retn
                             <1>
                             <1> cmp_cmd_show:
000079E0 B104                <1>       mov    cl, 4
000079E2 BF[720D0100]        <1>       mov    edi, Cmd_Show
000079E7 E87A030000          <1>       call   cmp_cmd
000079EC 0F83050A0000        <1>         jnc     show_file
                             <1>
                             <1> cmp_cmd_echo:
000079F2 B104                <1>       mov    cl, 4
000079F4 BF[AE0D0100]        <1>       mov    edi, Cmd_Echo
000079F9 E868030000          <1>       call   cmp_cmd
000079FE 7224                <1>       jc     short cmp_cmd_copy
                             <1>
                             <1>       ; 22/11/2017
                             <1>       ; AL = 0
00007A00 803E20              <1>       cmp    byte [esi], 20h
00007A03 7215                <1>       jb     short cmd_echo_nextline
                             <1>       ; 14/04/2016
00007A05 56                  <1>       push   esi
                             <1> cmd_echo_asciiz:
                             <1>       ;inc   esi
                             <1>       ;mov   al, [esi]
                             <1>       ; 22/11/2017
00007A06 AC                  <1>       lodsb
00007A07 3C20                <1>       cmp    al, 20h
00007A09 73FB                <1>       jnb    short cmd_echo_asciiz
```

```
00007A0B 4E                  <1>       dec    esi
00007A0C C60600              <1>       mov    byte [esi], 0
00007A0F 5E                  <1>       pop    esi
00007A10 89F7                <1>       mov    edi, esi
00007A12 E846E9FFFF          <1>       call   print_msg
00007A17 C60700              <1>       mov    byte [edi], 0
                             <1> cmd_echo_nextline:
00007A1A BE[B8190100]        <1>       mov    esi, NextLine
                             <1>       ;call  print_msg
                             <1>       ;retn
00007A1F E939E9FFFF          <1>       jmp    print_msg
                             <1>
                             <1> cmp_cmd_copy:
00007A24 B104                <1>       mov    cl, 4
00007A26 BF[950D0100]        <1>       mov    edi, Cmd_Copy
00007A2B E836030000          <1>       call   cmp_cmd
00007A30 0F83CC170000        <1>       jnc    copy_file
                             <1>
                             <1> cmp_cmd_move:
00007A36 B104                <1>       mov    cl, 4
00007A38 BF[9A0D0100]        <1>       mov    edi, Cmd_Move
00007A3D E824030000          <1>       call   cmp_cmd
00007A42 0F836E160000        <1>       jnc    move_file
                             <1>
                             <1> cmp_cmd_path:
00007A48 B104                <1>       mov    cl, 4
00007A4A BF[9F0D0100]        <1>       mov    edi, Cmd_Path
00007A4F E812030000          <1>       call   cmp_cmd
00007A54 0F83F0190000        <1>       jnc    set_get_path
                             <1>
                             <1> cmp_cmd_beep:
00007A5A B104                <1>       mov    cl, 4
00007A5C BF[CC0D0100]        <1>       mov    edi, Cmd_Beep
00007A61 E800030000          <1>       call   cmp_cmd
00007A66 720B                <1>       jc     short cmp_cmd_find
                             <1>       ; 13/05/2016
00007A68 8A3D[66580100]      <1>       mov    bh, [ptty] ; [ACTIVE_PAGE]
00007A6E E91FA3FFFF          <1>       jmp    beeper
                             <1>
                             <1> cmp_cmd_find:
00007A73 B104                <1>       mov    cl, 4
00007A75 BF[A90D0100]        <1>       mov    edi, Cmd_Find
00007A7A E8E7020000          <1>       call   cmp_cmd
00007A7F 0F82C4020000        <1>         jc      cmp_cmd_external
                             <1>
                             <1>       ;call  find_and_list_files
00007A85 E9AF220000          <1>       jmp    find_and_list_files
                             <1>       ;retn
                             <1>
                             <1> c_1:
00007A8A AD                  <1>       lodsd
                             <1> cmp_cmd_help:
00007A8B 3C3F                <1>       cmp    al, '?'
00007A8D 751D                <1>         jne     short cmp_cmd_remark
                             <1>
00007A8F BE[320D0100]        <1>       mov    esi, Command_List
                             <1> cmd_help_next_w:
00007A94 E8C4E8FFFF          <1>       call   print_msg
                             <1>
00007A99 803E20              <1>       cmp    byte [esi], 20h ; 0
00007A9C 7232                <1>       jb     short cmd_help_retn
                             <1>
00007A9E 56                  <1>       push   esi
00007A9F BE[6F190100]        <1>       mov    esi, nextline
00007AA4 E8B4E8FFFF          <1>       call   print_msg
00007AA9 5E                  <1>       pop    esi
00007AAA EBE8                <1>       jmp    short cmd_help_next_w
                             <1>
                             <1> cmp_cmd_remark:
00007AAC 3C2A                <1>       cmp    al, '*'
00007AAE 0F8595020000        <1>         jne     cmp_cmd_external
00007AB4 46                  <1>       inc    esi
00007AB5 BF[60590100]        <1>       mov    edi, Remark
00007ABA 8A06                <1>       mov    al, [esi]
00007ABC 3C20                <1>       cmp    al, 20h
00007ABE 7707                <1>       ja     short cmd_remark_write
00007AC0 89FE                <1>       mov    esi, edi ; Remark
00007AC2 E996E8FFFF          <1>       jmp    print_msg
                             <1>
                             <1> cmd_remark_write:
00007AC7 AA                  <1>       stosb
00007AC8 AC                  <1>       lodsb
00007AC9 3C20                <1>       cmp    al, 20h
00007ACB 73FA                <1>       jnb    short cmd_remark_write
00007ACD C60700              <1>       mov    byte [edi], 0
                             <1>
                             <1> cmd_help_retn:
                             <1> cmd_remark_retn:
                             <1> cd_retn:
00007AD0 C3                  <1>       retn
                             <1>
                             <1> c_2:
00007AD1 80F902              <1>       cmp    cl, 2
00007AD4 0F87AF000000        <1>         ja      c_3
00007ADA BE[AE590100]        <1>       mov    esi, CommandBuffer
00007ADF 72A9                <1>       jb     short c_1
                             <1>
                             <1> cmp_cmd_cd:
00007AE1 66AD                <1>       lodsw
00007AE3 663D4344            <1>       cmp    ax, 'CD'
00007AE7 7551                <1>       jne    short cmp_cmd_drive
00007AE9 46                  <1>         inc  esi
                             <1> cd_0:
```

```
00007AEA 668B06              <1>       mov    ax, [esi]
00007AED 3C20                <1>       cmp    al, 20h
00007AEF 76DF                <1>       jna    short cd_retn
                             <1>       ; 10/02/2016
00007AF1 80FC3A              <1>       cmp    ah, ':'
00007AF4 7504                <1>       jne    short cd_1
00007AF6 46                  <1>       inc    esi
00007AF7 46                  <1>       inc    esi
00007AF8 EB49                <1>       jmp    short cd_2
                             <1> 
                             <1> cd_1: ; change current directory
                             <1>       ; 29/11/2009
                             <1>       ; AH = CDh    ; to separate 'CD' command from others
                             <1>                     ; for restoring current directory
                             <1>                     ; 0CDh sign is for saving cdir into
                             <1>                     ; DOS drv description table cdir area
                             <1> 
00007AFA B4CD                <1>       mov    ah, 0CDh ; mov byte [CD_COMMAND], 0CDh
                             <1> 
00007AFC E81D230000          <1>       call   change_current_directory
00007B01 0F8337220000        <1>         jnc     change_prompt_dir_string
                             <1> 
                             <1> cd_error_messages:
00007B07 3C03                <1>       cmp    al, 3
00007B09 740C                <1>       je     short cd_path_not_found
                             <1>       ; 16/10/2016 (15h -> 15)
00007B0B 3C0F                <1>       cmp    al, 15 ; drive not ready error
00007B0D 7459                <1>       je     short cd_drive_not_ready
00007B0F 3C11                <1>       cmp    al, 17 ; read error
00007B11 7455                <1>       je     short cd_drive_not_ready
00007B13 3C13                <1>       cmp    al, 19 ; ; Bad directory/path name
00007B15 7466                <1>       je     short cd_command_failed
                             <1> 
                             <1> cd_path_not_found:
00007B17 50                  <1>       push   eax ; 29/12/2017
                             <1>       ;push  ax
00007B18 BE[D50F0100]        <1>       mov    esi, Msg_Dir_Not_Found
00007B1D E83BE8FFFF          <1>       call   print_msg
                             <1>       ;pop   ax
00007B22 58                  <1>       pop    eax ; 29/12/2017
00007B23 3A25[FC580100]      <1>       cmp    ah, [Current_Dir_Level]
00007B29 0F830F220000        <1>         jnb     change_prompt_dir_string
00007B2F 8825[FC580100]      <1>       mov    [Current_Dir_Level], ah
00007B35 E904220000          <1>         jmp     change_prompt_dir_string
                             <1> 
                             <1> cmp_cmd_drive: ; change current drive
                             <1>       ; C:, D:, E: etc.
00007B3A 80FC3A              <1>       cmp    ah, ':'
00007B3D 0F8506020000        <1>         jne     cmp_cmd_external
                             <1> 
                             <1> cd_2: ; 'CD C:', 'CD D:' ...
00007B43 803E20              <1>       cmp    byte [esi], 20h
00007B46 0F8707020000        <1>         ja      loc_cmd_failed
                             <1> 
00007B4C 24DF                <1>       and    al, 0DFh
00007B4E 2C41                <1>       sub    al, 'A'
00007B50 0F82FD010000        <1>         jc      loc_cmd_failed
                             <1> 
00007B56 3A05[D20C0100]      <1>         cmp     al, [Last_DOS_DiskNo]
00007B5C 770A                <1>         ja   short cd_drive_not_ready
                             <1> 
00007B5E 88C2                <1>       mov    dl, al
00007B60 E85BF3FFFF          <1>       call   change_current_drive
00007B65 7201                <1>       jc     short cd_drive_not_ready
00007B67 C3                  <1>       retn
                             <1> 
                             <1> cd_drive_not_ready:
00007B68 BE[920F0100]        <1>       mov    esi, Msg_Not_Ready_Read_Err
00007B6D E8EBE7FFFF          <1>       call   print_msg
                             <1> 
                             <1> cd_fail_drive_restart:
00007B72 8A15[FE580100]      <1>       mov    dl, [Current_Drv]
                             <1>       ;call  change_current_drive
00007B78 E943F3FFFF          <1>         jmp     change_current_drive
                             <1>       ;retn
                             <1> 
                             <1> cd_command_failed:
00007B7D BE[730F0100]        <1>       mov    esi, Msg_Bad_Command
00007B82 E8D6E7FFFF          <1>       call   print_msg
00007B87 EBE9                <1>       jmp    short cd_fail_drive_restart
                             <1> 
                             <1> c_3:
                             <1> cmp_cmd_dir:
00007B89 BF[320D0100]        <1>       mov    edi, Cmd_Dir
00007B8E E8D3010000          <1>       call   cmp_cmd
00007B93 0F8380020000        <1>       jnc    print_directory_list
                             <1> 
                             <1> cmp_cmd_cls:
00007B99 B103                <1>       mov    cl, 3
00007B9B BF[6E0D0100]        <1>       mov    edi, Cmd_Cls
00007BA0 E8C1010000          <1>       call   cmp_cmd
00007BA5 0F83C8E7FFFF        <1>         jnc clear_screen
                             <1> 
                             <1> cmp_cmd_ver:
00007BAB B103                <1>       mov    cl, 3
00007BAD BF[3C0D0100]        <1>       mov    edi, Cmd_Ver
00007BB2 E8AF010000          <1>       call   cmp_cmd
00007BB7 720A                <1>       jc     short cmp_cmd_mem
                             <1> 
00007BB9 BE[DA0C0100]        <1>       mov    esi, mainprog_Version
                             <1>       ;call  print_msg
00007BBE E99AE7FFFF          <1>       jmp    print_msg
                             <1>       ;retn
                             <1>
                             <1> cmp_cmd_mem:
00007BC3 B103                <1>       mov    cl, 3
00007BC5 BF[A40D0100]        <1>       mov    edi, Cmd_Mem
00007BCA E897010000          <1>       call   cmp_cmd
00007BCF 0F837FB6FFFF        <1>       jnc    memory_info
                             <1>
                             <1> cmp_cmd_del:
00007BD5 B103                <1>       mov    cl, 3
00007BD7 BF[770D0100]        <1>       mov    edi, Cmd_Del
00007BDC E885010000          <1>       call   cmp_cmd
00007BE1 0F83280F0000        <1>         jnc     delete_file
                             <1>
                             <1> cmp_cmd_set:
00007BE7 B103                <1>       mov    cl, 3
00007BE9 BF[6A0D0100]        <1>       mov    edi, Cmd_Set
00007BEE E873010000          <1>       call   cmp_cmd
00007BF3 0F83C9170000        <1>         jnc     set_get_env
                             <1>
                             <1> cmp_cmd_run:
00007BF9 B103                <1>       mov    cl, 3
00007BFB BF[660D0100]        <1>       mov    edi, Cmd_Run
00007C00 E861010000          <1>       call   cmp_cmd
                             <1>       ; 07/05/2016
00007C05 0F823E010000        <1>         jc      cmp_cmd_external
00007C0B E90F1E0000          <1>       jmp    load_and_execute_file
                             <1> c_5:
                             <1> cmp_cmd_mkdir:
00007C10 BF[8F0D0100]        <1>       mov    edi, Cmd_Mkdir
00007C15 E84C010000          <1>       call   cmp_cmd
00007C1A 0F83990A0000        <1>         jnc     make_directory
                             <1>
                             <1> cmp_cmd_rmdir:
00007C20 B105                <1>       mov    cl, 5
00007C22 BF[890D0100]        <1>       mov    edi, Cmd_Rmdir
00007C27 E83A010000          <1>       call   cmp_cmd
00007C2C 0F83AA0B0000        <1>         jnc     delete_directory
                             <1>
                             <1> cmp_cmd_chdir:
00007C32 B105                <1>       mov    cl, 5
00007C34 BF[C60D0100]        <1>       mov    edi, Cmd_Chdir
00007C39 E828010000          <1>       call   cmp_cmd
00007C3E 0F8205010000        <1>         jc      cmp_cmd_external
                             <1>
00007C44 E9A1FEFFFF          <1>       jmp    cd_0
                             <1>
                             <1> c_6:
00007C49 80F906              <1>       cmp    cl, 6
00007C4C 0F87E0000000        <1>         ja      c_8
00007C52 72BC                <1>       jb     short c_5
                             <1> cmp_cmd_prompt:
00007C54 BF[450D0100]        <1>       mov    edi, Cmd_Prompt
00007C59 E808010000          <1>       call   cmp_cmd
00007C5E 722F                <1>         jc   short cmp_cmd_volume
                             <1> get_prompt_name_fchar:
00007C60 AC                  <1>       lodsb
00007C61 3C20                <1>       cmp    al, 20h
00007C63 74FB                <1>       je     short get_prompt_name_fchar
00007C65 7713                <1>       ja     short loc_change_prompt_label
                             <1> default_command_prompt: ; 31/12/2017 ('sysprompt')
00007C67 BE[260D0100]        <1>       mov    esi, TRDOSPromptLabel
00007C6C C7065452444F        <1>       mov    dword [esi], "TRDO"
00007C72 66C746045300        <1>              mov    word [esi+4], "S"
                             <1> loc_cmd_prompt_return:
00007C78 C3                  <1>       retn
                             <1>
                             <1> set_command_prompt: ; 31/12/2017 ('sysprompt')
00007C79 AC                  <1>       lodsb
                             <1> loc_change_prompt_label:
00007C7A 66B90B00            <1>       mov    cx, 11
00007C7E BF[260D0100]        <1>       mov    edi, TRDOSPromptLabel
                             <1> put_char_new_prompt_label:
00007C83 AA                  <1>       stosb
00007C84 AC                  <1>       lodsb
00007C85 3C20                <1>       cmp    al, 20h
00007C87 7202                <1>       jb     short pass_put_new_prompt_label
00007C89 E2F8                <1>       loop   put_char_new_prompt_label
                             <1> pass_put_new_prompt_label:
00007C8B C60700              <1>       mov    byte [edi], 0
00007C8E C3                  <1>       retn
                             <1>
                             <1> cmp_cmd_volume:
00007C8F B106                <1>       mov    cl, 6
00007C91 BF[4C0D0100]        <1>       mov    edi, Cmd_Volume
00007C96 E8CB000000          <1>       call   cmp_cmd
00007C9B 7255                <1>         jc   short cmp_cmd_attrib
                             <1>
                             <1> cmd_vol1:
00007C9D AC                  <1>       lodsb
00007C9E 3C20                <1>       cmp    al, 20h
00007CA0 7707                <1>       ja     short cmd_vol2
00007CA2 A0[FE580100]        <1>       mov    al, [Current_Drv]
00007CA7 EB3D                <1>       jmp    short cmd_vol4
                             <1> cmd_vol2:
00007CA9 3C41                <1>       cmp    al, 'A'
00007CAB 0F82A2000000        <1>         jb      loc_cmd_failed
00007CB1 3C7A                <1>       cmp    al, 'z'
00007CB3 0F879A000000        <1>         ja      loc_cmd_failed
00007CB9 3C5A                <1>       cmp    al, 'Z'
00007CBB 760A                <1>       jna    short cmd_vol3
00007CBD 3C61                <1>       cmp    al, 'a'
00007CBF 0F828E000000        <1>         jb      loc_cmd_failed
00007CC5 24DF                <1>       and    al, 0DFh
```

```
                             <1> cmd_vol3:
00007CC7 8A26                <1>       mov    ah, [esi]
00007CC9 80FC3A              <1>       cmp    ah, ':'
00007CCC 0F8581000000        <1>         jne     loc_cmd_failed
00007CD2 2C41                <1>       sub    al, 'A'
00007CD4 3A05[D20C0100]      <1>         cmp     al, [Last_DOS_DiskNo]
00007CDA 760A                <1>       jna    short cmd_vol4
                             <1>
00007CDC BE[920F0100]        <1>       mov    esi, Msg_Not_Ready_Read_Err
00007CE1 E977E6FFFF          <1>       jmp    print_msg
                             <1>
                             <1> cmd_vol4:
00007CE6 E88EFAFFFF          <1>       call   print_volume_info
00007CEB 0F8277FEFFFF        <1>         jc      cd_drive_not_ready
00007CF1 C3                  <1>       retn
                             <1>
                             <1> cmp_cmd_attrib:
00007CF2 B106                <1>       mov    cl, 6
00007CF4 BF[7B0D0100]        <1>       mov    edi, Cmd_Attrib
00007CF9 E868000000          <1>       call   cmp_cmd
00007CFE 0F831D0F0000        <1>         jnc     set_file_attributes
                             <1>
                             <1> cmp_cmd_rename:
00007D04 B106                <1>       mov    cl, 6
00007D06 BF[820D0100]        <1>       mov    edi, Cmd_Rename
00007D0B E856000000          <1>       call   cmp_cmd
00007D10 0F8353110000        <1>         jnc     rename_file
                             <1>
                             <1> cmp_cmd_device:
00007D16 B106                <1>       mov    cl, 6
00007D18 BF[B70D0100]        <1>       mov    edi, Cmd_Device
00007D1D E844000000          <1>       call   cmp_cmd
00007D22 7225                <1>         jc   short cmp_cmd_external
                             <1>
00007D24 C3                  <1>       retn
                             <1>
                             <1> c_7:
                             <1> cmp_cmd_devlist:
00007D25 BF[BE0D0100]        <1>       mov    edi, Cmd_DevList
00007D2A E837000000          <1>       call   cmp_cmd
00007D2F 7218                <1>         jc   short cmp_cmd_external
                             <1>
                             <1> loc_cmd_return:
00007D31 C3                  <1>       retn
                             <1>
                             <1> c_8:
00007D32 80F908              <1>         cmp  cl, 8
00007D35 7712                <1>       ja     short cmp_cmd_external
00007D37 72EC                <1>       jb     short c_7
                             <1>
                             <1> cmp_cmd_longname:
00007D39 BF[530D0100]        <1>       mov    edi, Cmd_LongName
00007D3E E823000000          <1>       call   cmp_cmd
00007D43 0F8350060000        <1>         jnc     get_and_print_longname
                             <1>
                             <1> cmp_cmd_external:
                             <1>       ; 07/05/2016
                             <1>       ; 22/04/2016
00007D49 BE[AE590100]        <1>       mov    esi, CommandBuffer
00007D4E E9CC1C0000          <1>       jmp    loc_run_check_filename
                             <1>
                             <1> loc_cmd_failed:
00007D53 803D[AE590100]20    <1>       cmp    byte [CommandBuffer], 20h
00007D5A 76D5                <1>       jna    short loc_cmd_return
00007D5C BE[730F0100]        <1>       mov    esi, Msg_Bad_Command
                             <1> ;     call   print_msg
                             <1> ;loc_cmd_return:
                             <1> ;     retn
00007D61 E9F7E5FFFF          <1>       jmp    print_msg
                             <1>
                             <1> cmp_cmd:
                             <1>        ; 29/01/2016 (TRDOS 386 = TRDOS v2.0)
00007D66 BE[AE590100]        <1>          mov esi, CommandBuffer
                             <1>          ; edi = internal command word (ASCIIZ)
                             <1>        ; ecx = command length (<=8)
                             <1> cmp_cmd_1:
00007D6B AC                  <1>       lodsb
00007D6C AE                  <1>       scasb
00007D6D 750D                <1>       jne    short cmp_cmd_3
00007D6F E2FA                <1>       loop   cmp_cmd_1
00007D71 AC                  <1>       lodsb
00007D72 3C20                <1>       cmp    al, 20h
00007D74 7703                <1>       ja     short cmp_cmd_2
00007D76 30C0                <1>       xor    al, al
                             <1>       ; ZF = 1 -> internal command word matches
00007D78 C3                  <1>       retn
                             <1> cmp_cmd_2:
                             <1>       ; ZF = 0 (CF = 0) -> external command word
00007D79 58                  <1>       pop    eax ; no return to the caller from here
00007D7A EBCD                <1>       jmp    cmp_cmd_external
                             <1> cmp_cmd_3:
00007D7C F9                  <1>       stc
                             <1>       ; CF = 1 -> internal command word does not match
00007D7D C3                  <1>       retn
                             <1>
                             <1> loc_run_cmd_failed:
                             <1>       ; 15/03/2016
                             <1>       ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 07/12/2009 (CMD_INTR.ASM)
                             <1>       ; 29/11/2009
                             <1>
00007D7E E863000000          <1>       call   restore_cdir_after_cmd_fail
                             <1>
```

```
                              <1> loc_run_cmd_failed_cmp_al:
                              <1>       ; End of Restore_CDIR code (29/11/2009)
                              <1> 
00007D83 3C01                 <1>       cmp    al, 1 ; Bad command or file name
00007D85 74CC                 <1>       je     loc_cmd_failed
                              <1> loc_run_dir_not_found:
00007D87 3C03                 <1>       cmp    al, 3
00007D89 750A                 <1>       jne    short loc_run_file_notfound_msg
                              <1>       ; Path not found (MS-DOS Error Code = 3)
00007D8B BE[D50F0100]         <1>       mov    esi, Msg_Dir_Not_Found
00007D90 E9C8E5FFFF           <1>       jmp    print_msg
                              <1> 
                              <1> loc_run_file_notfound_msg:
00007D95 3C02                 <1>       cmp    al, 2 ; File not found
00007D97 750A                 <1>       jne    short loc_run_file_drv_read_err
                              <1> 
                              <1> loc_print_file_notfound_msg:
00007D99 BE[EC0F0100]         <1>         mov     esi, Msg_File_Not_Found
                              <1>       ;call  proc_printmsg
                              <1>       ;retn
00007D9E E9BAE5FFFF           <1>       jmp    print_msg
                              <1> 
                              <1> loc_run_file_drv_read_err:
                              <1>       ; Err: 17 (Read fault)
00007DA3 3C11                 <1>       cmp    al, 17 ; Drive not ready or read error
00007DA5 7404                 <1>       je     short loc_run_file_print_drv_read_err
                              <1>       ;
00007DA7 3C0F                 <1>       cmp    al, 15 ; Drive not ready (or read error)
00007DA9 750A                 <1>       jne    short loc_run_file_toobig
                              <1> 
                              <1> loc_run_file_print_drv_read_err:
00007DAB BE[920F0100]         <1>       mov    esi, Msg_Not_Ready_Read_Err
00007DB0 E9A8E5FFFF           <1>       jmp    print_msg
                              <1> 
                              <1> loc_run_file_toobig:
00007DB5 3C08                 <1>       cmp    al, 8 ; Not enough free memory to load&run file
00007DB7 750A                 <1>       jne    short loc_run_file_perm_denied
00007DB9 BE[37100100]         <1>       mov    esi, Msg_Insufficient_Memory
00007DBE E99AE5FFFF           <1>       jmp    print_msg
                              <1> 
                              <1> loc_run_file_perm_denied:
                              <1>       ; 29/12/2017
00007DC3 3C0B                 <1>       cmp    al, ERR_PERM_DENIED ; 11 ; Permission denied
00007DC5 750A                 <1>       jne    short loc_run_misc_error
00007DC7 BE[CC110100]         <1>       mov    esi, Msg_Permission_Denied
00007DCC E98CE5FFFF           <1>       jmp    print_msg
                              <1> 
                              <1>       ; 15/03/2016
                              <1> print_misc_error_msg:
                              <1> loc_run_misc_error:
                              <1>       ; AL = Error code
00007DD1 E8F3B4FFFF           <1>       call   bytetohex
00007DD6 66A3[6B100100]       <1>         mov     [error_code_hex], ax
                              <1> 
00007DDC BE[4E100100]         <1>       mov    esi, Msg_Error_Code
                              <1>       ;call  print_msg
                              <1>       ;retn
                              <1> 
00007DE1 E977E5FFFF           <1>       jmp    print_msg
                              <1> 
                              <1> restore_cdir_after_cmd_fail:
                              <1>       ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
00007DE6 50                   <1>       push   eax
00007DE7 8A3D[5E610100]       <1>       mov    bh, [RUN_CDRV] ; it is set at the beginning
                              <1>                             ; of the 'run' command.
00007DED 3A3D[FE580100]       <1>       cmp    bh, [Current_Drv]
00007DF3 7409                 <1>       je     short loc_run_restore_cdir
00007DF5 88FA                 <1>       mov    dl, bh
00007DF7 E8C4F0FFFF           <1>       call   change_current_drive
00007DFC EB19                 <1>       jmp    short loc_run_err_pass_restore_cdir
                              <1> 
                              <1> loc_run_restore_cdir:
00007DFE 803D[D30C0100]00     <1>       cmp    byte [Restore_CDIR], 0
00007E05 7610                 <1>       jna    short loc_run_err_pass_restore_cdir
00007E07 30DB                 <1>       xor    bl, bl
00007E09 0FB7F3               <1>       movzx  esi, bx
00007E0C 81C600010900         <1>       add    esi, Logical_DOSDisks
00007E12 E860F1FFFF           <1>       call   restore_current_directory
                              <1> 
                              <1> loc_run_err_pass_restore_cdir:
00007E17 58                   <1>       pop    eax
00007E18 C3                   <1>       retn
                              <1> 
                              <1> print_directory_list:
                              <1>       ; 10/02/2016
                              <1>       ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
                              <1>       ; 06/12/2009 ('cmp_cmd_dir')
                              <1>       ;
00007E19 66C705[A0620100]00-  <1>       mov    word [AttributesMask], 0800h ; ..except volume names..
00007E21 08                   <1>
00007E22 A0[FE580100]         <1>       mov    al, [Current_Drv]
00007E27 A2[5E610100]         <1>       mov    [RUN_CDRV], al
                              <1> get_dfname_fchar:
00007E2C AC                   <1>       lodsb
00007E2D 3C20                 <1>       cmp    al, 20h
00007E2F 74FB                 <1>       je     short get_dfname_fchar
00007E31 0F82A4000000         <1>         jb      loc_print_dir_call_all
00007E37 3C2D                 <1>       cmp    al, '-'
00007E39 7542                 <1>       jne    short loc_print_dir_call_flt
                              <1> get_next_attr_char:
00007E3B AC                   <1>       lodsb
00007E3C 3C20                 <1>       cmp    al, 20h
00007E3E 74FB                 <1>       je     short get_next_attr_char
```

```
00007E40 0F820DFFFFFF        <1>         jb      loc_cmd_failed
00007E46 24DF                <1>       and    al, 0DFh
00007E48 3C44                <1>       cmp    al, 'D' ; directories only ?
00007E4A 7512                <1>       jne    short pass_only_directories
00007E4C AC                  <1>       lodsb
00007E4D 3C20                <1>       cmp    al, 20h
00007E4F 0F87FEFEFFFF        <1>         ja      loc_cmd_failed
00007E55 800D[A0620100]10    <1>       or     byte [AttributesMask], 10h ; ..directory..
00007E5C EB18                <1>       jmp    short get_dfname_fchar_attr
                             <1> pass_only_directories:
00007E5E 3C46                <1>       cmp    al, 'F'      ; files only ?
00007E60 0F85B0000000        <1>         jne     check_attr_s
00007E66 AC                  <1>       lodsb
00007E67 3C20                <1>       cmp    al, 20h
00007E69 0F87E4FEFFFF        <1>         ja      loc_cmd_failed
00007E6F 800D[A1620100]10    <1>       or     byte [AttributesMask+1], 10h ; ..except directories..
                             <1> get_dfname_fchar_attr:
00007E76 AC                  <1>       lodsb
00007E77 3C20                <1>       cmp    al, 20h
00007E79 74FB                <1>       je     short get_dfname_fchar_attr
00007E7B 725E                <1>       jb     short loc_print_dir_call_all
                             <1>
                             <1> loc_print_dir_call_flt:
00007E7D 4E                  <1>       dec    esi
00007E7E BF[A2620100]        <1>       mov    edi, FindFile_Drv
00007E83 E8AC250000          <1>       call   parse_path_name
00007E88 7308                <1>       jnc    short loc_print_dir_change_drv_1
00007E8A 3C01                <1>       cmp    al, 1
00007E8C 0F87ECFEFFFF        <1>         ja      loc_run_cmd_failed
                             <1>
                             <1> loc_print_dir_change_drv_1:
00007E92 8A15[A2620100]      <1>       mov    dl, [FindFile_Drv]
                             <1> loc_print_dir_change_drv_2:
00007E98 3A15[5E610100]      <1>       cmp    dl, [RUN_CDRV]
00007E9E 740B                <1>       je     short loc_print_dir_change_directory
00007EA0 E81BF0FFFF          <1>       call   change_current_drive
00007EA5 0F82D3FEFFFF        <1>         jc      loc_run_cmd_failed
                             <1> loc_print_dir_change_directory:
00007EAB 803D[A3620100]20    <1>       cmp    byte [FindFile_Directory], 20h ; 0 or 20h ?
00007EB2 761D                <1>       jna    short pass_print_dir_change_directory
                             <1>
00007EB4 FE05[D30C0100]      <1>       inc    byte [Restore_CDIR]
00007EBA BE[A3620100]        <1>       mov    esi, FindFile_Directory
00007EBF 30E4                <1>       xor    ah, ah ; CD_COMMAND sign -> 0
00007EC1 E8581F0000          <1>       call   change_current_directory
00007EC6 0F82B2FEFFFF        <1>         jc      loc_run_cmd_failed
                             <1>
                             <1> loc_print_dir_change_prompt_dir_string:
00007ECC E86D1E0000          <1>       call   change_prompt_dir_string
                             <1>
                             <1> pass_print_dir_change_directory:
00007ED1 BE[E4620100]        <1>       mov    esi, FindFile_Name
00007ED6 803E20              <1>       cmp    byte [esi], 20h ; ; 0 or 20h ?
00007ED9 7706                <1>       ja     short loc_print_dir_call
                             <1>
                             <1> loc_print_dir_call_all:
00007EDB C7062A2E2A00        <1>       mov    dword [esi], '*.*'
                             <1> loc_print_dir_call:
00007EE1 E87E000000          <1>       call   print_directory
                             <1>
00007EE6 8A15[5E610100]      <1>       mov    dl, [RUN_CDRV]  ; it is set at the beginning
00007EEC 3A15[FE580100]      <1>       cmp    dl, [Current_Drv]
00007EF2 7406                <1>       je     short loc_print_dir_call_restore_cdir_retn
00007EF4 E8C7EFFFFF          <1>       call   change_current_drive
00007EF9 C3                  <1>       retn
                             <1>
                             <1> loc_print_dir_call_restore_cdir_retn:
00007EFA 803D[D30C0100]00    <1>       cmp    byte [Restore_CDIR], 0
00007F01 7610                <1>       jna    short pass_print_dir_call_restore_cdir_retn
                             <1>
00007F03 BE00010900          <1>       mov    esi, Logical_DOSDisks
00007F08 31C0                <1>       xor    eax, eax
00007F0A 88D4                <1>       mov    ah, dl
00007F0C 01C6                <1>       add    esi, eax
                             <1>
00007F0E E864F0FFFF          <1>       call   restore_current_directory
                             <1>
                             <1> pass_print_dir_call_restore_cdir_retn:
00007F13 C3                  <1>       retn
                             <1>
                             <1> check_attr_s_cap:
00007F14 24DF                <1>       and    al, 0DFh
                             <1> check_attr_s:
00007F16 3C53                <1>       cmp    al, 'S'
00007F18 7514                <1>       jne    short pass_attr_s
00007F1A 800D[A0620100]04    <1>       or     byte [AttributesMask], 4 ; system
00007F21 AC                  <1>       lodsb
00007F22 3C20                <1>       cmp    al, 20h
00007F24 0F844CFFFFFF        <1>         je      get_dfname_fchar_attr
00007F2A 72AF                <1>       jb     short loc_print_dir_call_all
00007F2C 24DF                <1>       and    al, 0DFh
                             <1> pass_attr_s:
00007F2E 3C48                <1>       cmp    al, 'H'
00007F30 7514                <1>       jne    short pass_attr_h
00007F32 800D[A0620100]02    <1>       or     byte [AttributesMask], 2 ; hidden
                             <1> pass_attr_shr:
00007F39 AC                  <1>       lodsb
00007F3A 3C20                <1>       cmp    al, 20h
00007F3C 0F8434FFFFFF        <1>         je      get_dfname_fchar_attr
00007F42 7297                <1>       jb     short loc_print_dir_call_all
00007F44 EBCE                <1>       jmp    short check_attr_s_cap
                             <1>
                             <1> pass_attr_h:
```

```
00007F46 3C52                <1>       cmp    al, 'R'
00007F48 7509                <1>       jne    short pass_attr_r
00007F4A 800D[A0620100]01    <1>       or     byte [AttributesMask], 1 ; read only
00007F51 EBE6                <1>       jmp    short pass_attr_shr
                             <1>
                             <1> pass_attr_r:
00007F53 3C41                <1>       cmp    al, 'A'
00007F55 0F85F8FDFFFF        <1>         jne     loc_cmd_failed
00007F5B 800D[A0620100]20    <1>       or     byte [AttributesMask], 20h ; archive
00007F62 EBD5                <1>       jmp    short pass_attr_shr
                             <1>
                             <1> print_directory:
                             <1>       ; 13/05/2016
                             <1>       ; 11/02/2016
                             <1>       ; 10/02/2016
                             <1>       ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 30/10/2010 ('proc_print_directory')
                             <1>       ; 19/09/2009
                             <1>       ; 2005
                             <1>       ; INPUT ->
                             <1>       ;      ESI = Asciiz File/Dir Name Address
                             <1>
00007F64 56                  <1>       push   esi
                             <1>
00007F65 29C0                <1>       sub    eax, eax
                             <1>
00007F67 66A3[2C630100]      <1>       mov    word [Dir_Count], ax ; 0
00007F6D 66A3[2A630100]      <1>       mov    word [File_Count], ax ; 0
00007F73 A3[2E630100]        <1>       mov    dword [Total_FSize], eax ; 0
                             <1>
00007F78 E8F6E3FFFF          <1>       call    clear_screen
                             <1>
00007F7D 31C9                <1>       xor    ecx, ecx
00007F7F 8A2D[FE580100]      <1>       mov     ch, [Current_Drv] ; DirBuff_Drv - 'A'
00007F85 A0[FF580100]        <1>       mov     al, [Current_Dir_Drv]
00007F8A A2[900E0100]        <1>       mov     [Dir_Drive_Name], al
00007F8F BE00010900          <1>       mov    esi, Logical_DOSDisks
00007F94 01CE                <1>       add    esi, ecx
                             <1>
00007F96 E858F9FFFF          <1>       call   move_volume_name_and_serial_no
00007F9B 730C                <1>       jnc    short print_dir_strlen_check
                             <1>
00007F9D 5E                  <1>       pop    esi
00007F9E 8A3D[66580100]      <1>       mov    bh, [ptty] ; [ACTIVE_PAGE]
                             <1>       ;call  beeper
                             <1>       ;retn
00007FA4 E9E99DFFFF          <1>       jmp    beeper  ; beep ! and return
                             <1>
                             <1> print_dir_strlen_check:
00007FA9 BE[01590100]        <1>       mov    esi, Current_Dir_Root
00007FAE BF[2D0F0100]        <1>       mov    edi, Dir_Str_Root
                             <1>
                             <1>       ;xor   ecx, ecx
00007FB3 8A0D[5D590100]      <1>         mov     cl, [Current_Dir_StrLen]
00007FB9 FEC1                <1>       inc    cl
00007FBB 80F940              <1>       cmp    cl, 64
00007FBE 760D                <1>       jna    short pass_print_dir_strlen_shorting
00007FC0 46                  <1>       inc    esi
00007FC1 01CE                <1>       add    esi, ecx
00007FC3 83EE40              <1>       sub    esi, 64
00007FC6 47                  <1>       inc    edi
00007FC7 B82E2E2E20          <1>       mov    eax, '... '
00007FCC AB                  <1>       stosd
                             <1>
                             <1> pass_print_dir_strlen_shorting:
00007FCD F3A4                <1>       rep    movsb
                             <1>
00007FCF BE[830E0100]        <1>       mov    esi, Dir_Drive_Str
00007FD4 E884E3FFFF          <1>       call   print_msg
                             <1>
00007FD9 BE[E20E0100]        <1>       mov    esi, Vol_Serial_Header
00007FDE E87AE3FFFF          <1>       call   print_msg
                             <1>
00007FE3 BE[220F0100]        <1>       mov    esi, Dir_Str_Header
00007FE8 E870E3FFFF          <1>       call   print_msg
                             <1>
00007FED BE[6D190100]        <1>       mov    esi, next2line
00007FF2 E866E3FFFF          <1>       call   print_msg
                             <1>
                             <1> loc_print_dir_first_file:
00007FF7 C605[41630100]10    <1>       mov    byte [PrintDir_RowCounter], 16
00007FFE 66A1[A0620100]      <1>       mov    ax, [AttributesMask]
00008004 5E                  <1>       pop    esi
                             <1>
00008005 E859020000          <1>       call   find_first_file
0000800A 0F826F010000        <1>         jc      loc_dir_ok
                             <1>
                             <1> loc_dfname_use_this:
                             <1>       ; bl = File Attributes (bh = Long Name Entry Length)
00008010 F6C310              <1>       test   bl, 10h  ; Is it a directory?
00008013 741B                <1>       jz     short loc_not_dir
                             <1>
00008015 66FF05[2C630100]    <1>       inc    word [Dir_Count]
0000801C 89F2                <1>       mov    edx, esi      ; FindFile_DirEntry address
0000801E BE[72100100]        <1>       mov    esi, Type_Dir; '<DIR>      '
00008023 BF[89100100]        <1>       mov    edi, Dir_Or_FileSize
                             <1>       ; move 10 bytes
00008028 A5                  <1>       movsd
00008029 A5                  <1>       movsd
0000802A 66A5                <1>       movsw
0000802C 89D6                <1>       mov    esi, edx
0000802E EB36                <1>       jmp     short loc_dir_attribute
                             <1>
                             <1> loc_not_dir:
00008030 66FF05[2A630100]    <1>       inc    word [File_Count]
00008037 0105[2E630100]      <1>       add    [Total_FSize], eax
                             <1> 
0000803D B90A000000          <1>       mov    ecx, 10  ; 32 bit divisor
00008042 89CF                <1>       mov    edi, ecx
00008044 81C7[89100100]      <1>       add    edi, Dir_Or_FileSize
                             <1> loc_dir_rdivide:
0000804A 29D2                <1>       sub    edx, edx
0000804C F7F1                <1>       div    ecx     ; remainder in dl (< 10)
0000804E 80C230              <1>       add     dl, '0'      ; to make visible (ascii)
00008051 4F                  <1>       dec    edi
00008052 8817                <1>       mov     [edi], dl
00008054 21C0                <1>       and    eax, eax
00008056 75F2                <1>       jnz    short loc_dir_rdivide
                             <1> 
                             <1> loc_dir_fill_space:
00008058 81FF[89100100]      <1>       cmp     edi, Dir_Or_FileSize
0000805E 7606                <1>       jna     short loc_dir_attribute
00008060 4F                  <1>       dec     edi
00008061 C60720              <1>       mov     byte [edi], 20h
00008064 EBF2                <1>       jmp     short loc_dir_fill_space
                             <1> 
                             <1> loc_dir_attribute:
00008066 C705[94100100]2020- <1>       mov    dword [File_Attribute], 20202020h
0000806E 2020                <1> 
                             <1> 
00008070 80FB20              <1>       cmp    bl, 20h  ; Is it an archive file?
00008073 7207                <1>       jb     short loc_dir_pass_arch
00008075 C605[97100100]41    <1>       mov    byte [File_Attribute+3], 'A'
                             <1> 
                             <1> loc_dir_pass_arch:
0000807C 80E307              <1>       and    bl, 7
0000807F 7428                <1>       jz     short loc_dir_file_name
00008081 88DF                <1>       mov    bh, bl
00008083 80E303              <1>       and    bl, 3
00008086 38DF                <1>       cmp    bh, bl
00008088 7607                <1>       jna    short loc_dir_pass_s
0000808A C605[94100100]53    <1>       mov    byte [File_Attribute], 'S'
                             <1> 
                             <1> loc_dir_pass_s:
00008091 80E302              <1>       and     bl,2
00008094 7407                <1>       jz      short loc_dir_pass_h
00008096 C605[95100100]48    <1>       mov     byte [File_Attribute+1], 'H'
                             <1> loc_dir_pass_h:
0000809D 80E701              <1>       and     bh,1
000080A0 7407                <1>       jz      short loc_dir_file_name
000080A2 C605[96100100]52    <1>       mov     byte [File_Attribute+2], 'R'
                             <1> loc_dir_file_name:
                             <1>       ;mov     bx, [esi+18h] ; Date
                             <1>       ;mov     dx, [esi+16h] ; Time
000080A9 8B5E16              <1>       mov    ebx, [esi+16h]
000080AC 89F1                <1>       mov    ecx, esi ; FindFile_DirEntry address
000080AE BF[7C100100]        <1>       mov     edi, File_Name
                             <1>       ; move 8 bytes
000080B3 A5                  <1>       movsd
000080B4 A5                  <1>       movsd
000080B5 C60720              <1>       mov    byte [edi], 20h
000080B8 47                  <1>       inc    edi
                             <1>       ; move 3 bytes
000080B9 66A5                <1>       movsw
000080BB A4                  <1>       movsb
000080BC 89CE                <1>       mov    esi, ecx
                             <1> 
                             <1> Dir_Time_start:
                             <1>       ;mov   ax, dx       ; Time
000080BE 6689D8              <1>       mov    ax, bx
000080C1 66C1E805            <1>       shr    ax, 5        ; shift right 5 times
000080C5 6683E03F            <1>       and    ax, 0000111111b     ; Minute Mask
000080C9 D40A                <1>       aam                 ; Q([AL]/10)->AH
                             <1>                           ; R([AL]/10)->AL
                             <1>                           ; [AL]+[AH]= Minute as BCD
000080CB 660D3030            <1>       or     ax, '00'     ; Convert to ASCII
000080CF 86E0                <1>       xchg   ah, al
000080D1 66A3[A7100100]      <1>       mov    [File_Minute], ax
                             <1> 
                             <1>       ;mov   al, dh
000080D7 88F8                <1>       mov    al, bh
000080D9 C0E803              <1>       shr    al, 3        ; shift right 3 times
000080DC D40A                <1>       aam                 ; [AL]+[AH]= Hours as BCD
000080DE 660D3030            <1>       or     ax, '00'
000080E2 86E0                <1>       xchg   ah, al
000080E4 66A3[A4100100]      <1>       mov     [File_Hour], ax
                             <1> 
000080EA C1EB10              <1>       shr    ebx, 16             ; BX = Date
                             <1> 
                             <1> Dir_Date_start:
000080ED 6689D8              <1>       mov    ax, bx       ; Date
000080F0 6683E01F            <1>       and    ax, 00011111b; Day Mask
000080F4 D40A                <1>       aam                 ; Q([AL]/10)->AH
                             <1>                           ; R([AL]/10)->AL
                             <1>                           ; [AL]+[AH]= Day as BCD
000080F6 660D3030            <1>       or     ax, '00'     ; Convert to ASCII
000080FA 86C4                <1>       xchg   al, ah
                             <1> 
000080FC 66A3[99100100]      <1>       mov    [File_Day], ax
                             <1> 
00008102 6689D8              <1>       mov    ax, bx
00008105 66C1E805            <1>       shr    ax, 5        ; shift right 5 times
00008109 6683E00F            <1>       and    ax, 00001111b; Month Mask
0000810D D40A                <1>       aam
0000810F 660D3030            <1>       or     ax, '00'
00008113 86E0                <1>       xchg   ah, al
```

```
00008115 66A3[9C100100]      <1>       mov    [File_Month], ax
                             <1>
0000811B 6689D8              <1>       mov    ax, bx
0000811E 66C1E809            <1>       shr     ax, 9
00008122 6683E07F            <1>       and    ax, 01111111b; Result = Year - 1980
00008126 6605BC07            <1>       add    ax, 1980
                             <1>
0000812A B10A                <1>       mov    cl, 10
0000812C F6F1                <1>       div    cl            ; Q -> AL, R -> AH
0000812E 80CC30              <1>       or     ah, '0'
00008131 8825[A2100100]      <1>       mov    [File_Year+3], ah
00008137 D40A                <1>       aam
00008139 86E0                <1>       xchg   ah, al
0000813B 80CC30              <1>       or     ah, '0'         ; Convert to ASCII
0000813E 8825[A1100100]      <1>       mov    [File_Year+2], ah
00008144 D40A                <1>       aam
00008146 86C4                <1>       xchg   al, ah
00008148 660D3030            <1>       or     ax, '00'
0000814C 66A3[9F100100]      <1>       mov    [File_Year], ax
                             <1>
                             <1> loc_show_line:
00008152 56                  <1>       push   esi
00008153 BE[7C100100]        <1>       mov     esi, File_Name
00008158 E800E2FFFF          <1>       call   print_msg
0000815D BE[6F190100]        <1>       mov    esi, nextline
00008162 E8F6E1FFFF          <1>       call   print_msg
00008167 5E                  <1>       pop    esi
                             <1>
00008168 FE0D[41630100]      <1>       dec    byte [PrintDir_RowCounter]
0000816E 0F84D4000000        <1>         jz      pause_dir_scroll
                             <1>
                             <1> loc_next_entry:
00008174 E899010000          <1>       call   find_next_file
00008179 0F8391FEFFFF        <1>         jnc     loc_dfname_use_this
                             <1>
                             <1> loc_dir_ok:
0000817F B90A000000          <1>       mov     ecx, 10
00008184 66A1[2C630100]      <1>       mov    ax, [Dir_Count]
0000818A BF[BD100100]        <1>       mov    edi, Decimal_Dir_Count
0000818F 6639C8              <1>       cmp    ax, cx ; 10
00008192 7216                <1>       jb     short pass_ddc
00008194 47                  <1>       inc    edi
00008195 6683F864            <1>       cmp    ax, 100
00008199 720F                <1>       jb     short pass_ddc
0000819B 47                  <1>       inc    edi
0000819C 663DE803            <1>       cmp    ax, 1000
000081A0 7208                <1>       jb     short pass_ddc
000081A2 47                  <1>       inc    edi
000081A3 663D1027            <1>       cmp    ax, 10000
000081A7 7201                <1>       jb     short pass_ddc
000081A9 47                  <1>       inc    edi
                             <1> pass_ddc:
000081AA 886F01              <1>       mov     [edi+1], ch ; 0
                             <1> loc_ddc_rediv:
000081AD 31D2                <1>       xor     edx, edx
000081AF 66F7F1              <1>       div     cx   ; 10
000081B2 80C230              <1>       add     dl, '0'
000081B5 8817                <1>       mov     [edi], dl
000081B7 4F                  <1>       dec     edi
000081B8 6609C0              <1>       or     ax, ax
000081BB 75F0                <1>       jnz    short loc_ddc_rediv
                             <1>
000081BD 66A1[2A630100]      <1>       mov     ax, [File_Count]
000081C3 BF[AC100100]        <1>       mov     edi, Decimal_File_Count
000081C8 6639C8              <1>       cmp     ax, cx ; 10
000081CB 7216                <1>       jb      short pass_dfc
000081CD 47                  <1>       inc     edi
000081CE 6683F864            <1>       cmp     ax, 100
000081D2 720F                <1>       jb      short pass_dfc
000081D4 47                  <1>       inc     edi
000081D5 663DE803            <1>       cmp     ax, 1000
000081D9 7208                <1>       jb      short pass_dfc
000081DB 47                  <1>       inc     edi
000081DC 663D1027            <1>       cmp     ax, 10000
000081E0 7201                <1>       jb      short pass_dfc
000081E2 47                  <1>       inc     edi
                             <1> pass_dfc:
                             <1>       ;mov    cx, 10
000081E3 886F01              <1>       mov     [edi+1], ch ; 00
                             <1> loc_dfc_rediv:
                             <1>       ;xor   dx, dx
000081E6 30D2                <1>       xor    dl, dl
000081E8 66F7F1              <1>       div    cx
000081EB 80C230              <1>       add    dl, '0'
000081EE 8817                <1>       mov    [edi], dl
000081F0 4F                  <1>       dec    edi
000081F1 6609C0              <1>       or     ax, ax
000081F4 75F0                <1>       jnz    short loc_dfc_rediv
                             <1>
000081F6 BF[40630100]        <1>       mov     edi, TFS_Dec_End
                             <1>         ;mov    byte [edi], 0
000081FB A1[2E630100]        <1>       mov     eax, [Total_FSize]
                             <1>       ;mov    ecx, 10
                             <1> rediv_tfs_hex:
                             <1>       ;sub   edx, edx
00008200 28D2                <1>       sub    dl, dl
00008202 F7F1                <1>       div    ecx
00008204 80C230              <1>       add    dl, '0'
00008207 4F                  <1>       dec     edi
00008208 8817                <1>       mov     [edi], dl
0000820A 21C0                <1>       and    eax, eax
0000820C 75F2                <1>       jnz    short rediv_tfs_hex
                             <1>
```

```
0000820E 893D[32630100]      <1>       mov    [TFS_Dec_Begin], edi
00008214 BE[AA100100]        <1>       mov    esi, Decimal_File_Count_Header
00008219 E83FE1FFFF          <1>       call   print_msg
0000821E BE[B2100100]        <1>       mov    esi, str_files
00008223 E835E1FFFF          <1>       call   print_msg
00008228 BE[C3100100]        <1>       mov    esi, str_dirs
0000822D E82BE1FFFF          <1>       call   print_msg
00008232 8B35[32630100]      <1>       mov    esi, [TFS_Dec_Begin]
00008238 E820E1FFFF          <1>       call   print_msg
0000823D BE[D4100100]        <1>       mov    esi, str_bytes
00008242 E816E1FFFF          <1>       call   print_msg
                             <1> 
00008247 C3                  <1>       retn
                             <1> 
                             <1> pause_dir_scroll:
00008248 28E4                <1>       sub    ah, ah
0000824A E8C789FFFF          <1>       call   int16h
0000824F 3C1B                <1>       cmp    al, 1Bh
00008251 0F8428FFFFFF        <1>         je      loc_dir_ok
00008257 C605[41630100]10    <1>       mov    byte [PrintDir_RowCounter], 16 ; Reset counter
0000825E E911FFFFFF          <1>         jmp     loc_next_entry
                             <1> 
                             <1> find_first_file:
                             <1>       ; 11/02/2016
                             <1>       ; 10/02/2016
                             <1>       ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 09/10/2011
                             <1>       ; 17/09/2009
                             <1>       ; 2005
                             <1>       ; INPUT ->
                             <1>       ;      ESI = ASCIIZ File/Dir Name Address (in Current Directory)
                             <1>       ;      AL = Attributes AND mask (The AND result must be equal to AL)
                             <1>       ;            bit 0 = Read Only
                             <1>       ;            bir 1 = Hidden
                             <1>       ;            bit 2 = System
                             <1>       ;            bit 3 = Volume Label
                             <1>       ;            bit 4 = Directory
                             <1>       ;            bit 5 = Archive
                             <1>       ;            bit 6 = Reserved, must be 0
                             <1>       ;            bit 7 = Reserved, must be 0
                             <1>       ;        AH = Attributes Negative AND mask (The AND result must be ZERO)
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      CF = 1 -> Error, Error Code in EAX (AL)
                             <1>       ;      CF = 0 ->
                             <1>       ;           ESI = Directory Entry (FindFile_DirEntry) Location
                             <1>       ;           EDI = Directory Buffer Directory Entry Location
                             <1>       ;           EAX = File Size
                             <1>       ;            BL = Attributes of The File/Directory
                             <1>       ;            BH = Long Name Yes/No Status (>0 is YES)
                             <1>       ;              DX > 0 : Ambiguous filename chars are used
                             <1>       ;
                             <1>       ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
                             <1> 
00008263 66A3[F2620100]      <1>       mov    [FindFile_AttributesMask], ax
00008269 BF[F4620100]        <1>       mov    edi, FindFile_DirEntry ; TR-DOS Fullfilename formatted buffer
0000826E 31C0                <1>       xor    eax, eax
00008270 B90B000000          <1>       mov    ecx, 11
00008275 F3AB                <1>       rep    stosd  ; 44 bytes
                             <1>       ;stosw        ; +2 bytes
                             <1> 
00008277 BF[E4620100]        <1>       mov    edi, FindFile_Name ; FFF structure, offset 66
0000827C 39FE                <1>       cmp    esi, edi
0000827E 7408                <1>       je     short loc_fff_mfn_ok
00008280 89FA                <1>       mov    edx, edi
                             <1>        ; move 13 bytes
00008282 A5                  <1>       movsd
00008283 A5                  <1>       movsd
00008284 A5                  <1>       movsd
00008285 AA                  <1>       stosb
00008286 89D6                <1>       mov    esi, edx
                             <1> loc_fff_mfn_ok:
00008288 BF[93620100]        <1>       mov    edi, Dir_Entry_Name ; Dir Entry Format File Name
0000828D E8D7200000          <1>       call   convert_file_name
00008292 89FE                <1>       mov    esi, edi ; offset Dir_Entry_Name
                             <1> 
00008294 66A1[F2620100]      <1>       mov    ax, [FindFile_AttributesMask]
                             <1>       ;xor   ecx, ecx
0000829A 30C9                <1>       xor    cl, cl
0000829C E8D11D0000          <1>       call   locate_current_dir_file
000082A1 726E                <1>       jc     short loc_fff_retn
                             <1>       ; EDI = Directory Entry
                             <1>       ; EBX = Directory Buffer Entry Index/Number
                             <1> 
                             <1> loc_fff_fnf_ln_check:
000082A3 30ED                <1>       xor    ch, ch
000082A5 80F60F              <1>       xor    dh, 0Fh
000082A8 7408                <1>       jz     short loc_fff_longname_yes
000082AA 882D[F1620100]      <1>       mov    [FindFile_LongNameYes], ch ; 0
000082B0 EB0C                <1>       jmp    short loc_fff_longname_no
                             <1> 
                             <1> loc_fff_longname_yes:
                             <1>       ;inc   byte [FindFile_LongNameYes]
000082B2 8A0D[FE610100]      <1>       mov    cl, [LFN_EntryLength]
000082B8 880D[F1620100]      <1>       mov    [FindFile_LongNameEntryLength], cl ; FindFile_LongNameYes
                             <1> 
                             <1> loc_fff_longname_no:
                             <1>       ;mov   bx, [DirBuff_CurrentEntry]
000082BE 66891D[1C630100]    <1>       mov    [FindFile_DirEntryNumber], bx
000082C5 6689C2              <1>       mov    dx, ax ; Ambiguous Filename chars used sign > 0
                             <1> 
000082C8 A0[FE580100]        <1>       mov    al, [Current_Drv]
000082CD A2[A2620100]        <1>       mov    [FindFile_Drv], al
```

```
                             <1>
000082D2 A1[F8580100]        <1>       mov    eax, [Current_Dir_FCluster]
000082D7 A3[14630100]        <1>       mov    [FindFile_DirFirstCluster], eax
                             <1>
000082DC A1[2D610100]        <1>       mov    eax, [DirBuff_Cluster]
000082E1 A3[18630100]        <1>       mov    [FindFile_DirCluster], eax
                             <1>
000082E6 66FF05[1E630100]    <1>       inc    word [FindFile_MatchCounter]
                             <1>
000082ED 89FB                <1>       mov    ebx, edi
000082EF 89FE                <1>       mov    esi, edi
000082F1 BF[F4620100]        <1>       mov    edi, FindFile_DirEntry
000082F6 89F8                <1>       mov    eax, edi
000082F8 B108                <1>       mov    cl, 8
000082FA F3A5                <1>       rep    movsd
000082FC 89C6                <1>       mov    esi, eax
000082FE 89DF                <1>       mov    edi, ebx
                             <1>
00008300 A1[10630100]        <1>       mov    eax, [FindFile_DirEntry+28] ; File Size
                             <1>
00008305 8A1D[FF620100]      <1>       mov    bl, [FindFile_DirEntry+11] ; File Attributes
0000830B 8A3D[F1620100]      <1>       mov    bh, [FindFile_LongNameYes]
                             <1>
                             <1>       ;mov   cx, [DirBuff_EntryCounter]
                             <1>       ;mov   [FindFile_DirEntryNumber], cx
                             <1>       ;mov   cx, [FindFile_DirEntryNumber]
                             <1>       ; ecx = 0
                             <1>
                             <1> loc_fff_retn:
00008311 C3                  <1>       retn
                             <1>
                             <1> find_next_file:
                             <1>       ; 15/10/2016
                             <1>       ; 10/02/2016
                             <1>       ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 06/02/2011
                             <1>       ; 17/09/2009
                             <1>       ; 2005
                             <1>       ; INPUT ->
                             <1>       ;      NONE, Find First File Parameters
                             <1>       ; OUTPUT ->
                             <1>       ;      CF = 1 -> Error, Error Code in EAX (AL)
                             <1>       ;      CF = 0 ->
                             <1>       ;          ESI = Directory Entry (FindFile_DirEntry) Location
                             <1>       ;          EDI = Directory Buffer Directory Entry Location
                             <1>       ;          EAX = File Size
                             <1>       ;            BL = Attributes of The File/Directory
                             <1>       ;            BH = Long Name Yes/No Status (>0 is YES)
                             <1>       ;             DX > 0 : Ambiguous filename chars are used
                             <1>       ;
                             <1>       ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
                             <1>
00008312 66833D[1E630100]00  <1>       cmp    word [FindFile_MatchCounter], 0
0000831A 7707                <1>       ja     short loc_start_search_next_file
                             <1>
                             <1> loc_fnf_stc_retn:
0000831C F9                  <1>       stc
                             <1> loc_fnf_ax12h_retn:
0000831D B80C000000          <1>       mov    eax, 12 ; No More files
                             <1> ;loc_fnf_retn:
00008322 C3                  <1>       retn
                             <1>
                             <1> loc_start_search_next_file:
00008323 668B1D[1C630100]    <1>       mov    bx, [FindFile_DirEntryNumber]
0000832A 6643                <1>       inc    bx
0000832C 663B1D[2B610100]    <1>       cmp    bx, [DirBuff_LastEntry]
00008333 7719                <1>       ja     short loc_cont_search_next_file
                             <1>
                             <1> loc_fnf_search:
00008335 BE[93620100]        <1>       mov    esi, Dir_Entry_Name
0000833A 66A1[F2620100]      <1>       mov    ax, [FindFile_AttributesMask]
00008340 6631C9              <1>       xor    cx, cx
00008343 E82E1E0000          <1>       call   find_directory_entry
00008348 0F8355FFFFFF        <1>         jnc     loc_fff_fnf_ln_check
                             <1>
                             <1> loc_cont_search_next_file:
0000834E 31DB                <1>       xor    ebx, ebx
00008350 8A3D[FE580100]      <1>       mov    bh, [Current_Drv]
00008356 BE00010900          <1>       mov    esi, Logical_DOSDisks
0000835B 01DE                <1>       add    esi, ebx
                             <1>
0000835D 803D[FC580100]00    <1>       cmp    byte [Current_Dir_Level], 0
00008364 7608                <1>       jna    short loc_fnf_check_FAT_type
00008366 807E0301            <1>       cmp    byte [esi+LD_FATType], 1
0000836A 72B1                <1>       jb     short loc_fnf_ax12h_retn
0000836C EB06                <1>       jmp    short loc_fnf_check_next_cluster
                             <1>
                             <1> loc_fnf_check_FAT_type:
0000836E 807E0303            <1>       cmp    byte [esi+LD_FATType], 3
00008372 72A9                <1>       jb     short loc_fnf_ax12h_retn
                             <1>
                             <1> loc_fnf_check_next_cluster:
00008374 A1[2D610100]        <1>       mov    eax, [DirBuff_Cluster]
00008379 E8CA370000          <1>       call   get_next_cluster
0000837E 7306                <1>       jnc    short loc_fnf_load_next_dir_cluster
00008380 09C0                <1>       or     eax, eax
00008382 7498                <1>       jz     short loc_fnf_stc_retn
                             <1>       ;mov   eax, 17 ;Drive not ready or read error
00008384 F5                  <1>       cmc    ;stc
                             <1> loc_fnf_retn:
00008385 C3                  <1>       retn
                             <1>
                             <1> loc_fnf_load_next_dir_cluster:
```

```
00008386 E8A3390000          <1>       call   load_FAT_sub_directory
0000838B 72F8                <1>       jc     short loc_fnf_retn
0000838D 6631DB              <1>       xor    bx, bx
00008390 66891D[1C630100]    <1>       mov    [FindFile_DirEntryNumber], bx
00008397 EB9C                <1>       jmp    short loc_fnf_search
                             <1> 
                             <1> get_and_print_longname:
                             <1>       ; 16/10/2016
                             <1>       ; 13/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 24/01/2010
                             <1>       ; 17/10/2009 (CMD_INTR.ASM, 'cmp_cmd_longname')
                             <1> get_longname_fchar:
00008399 803E20              <1>       cmp    byte [esi], 20h
0000839C 7701                <1>       ja     short loc_find_longname
                             <1>       ;jb    short loc_longname_retn
                             <1>       ;inc   esi
                             <1>       ;je    short get_longname_fchar
                             <1> ;loc_longname_retn:
0000839E C3                  <1>       retn
                             <1> loc_find_longname:
0000839F E839210000          <1>       call   find_longname
000083A4 7328                <1>       jnc    short loc_print_longname
                             <1> 
000083A6 08C0                <1>       or     al, al
000083A8 741A                <1>       jz     short loc_longname_not_found
                             <1> 
                             <1>       ; 16/10/2016 (15h -> 15, 17)
000083AA 3C0F                <1>       cmp    al, 15
000083AC 0F84B6F7FFFF        <1>       je     cd_drive_not_ready ; drive not ready
                             <1>                              ; or
000083B2 3C11                <1>       cmp    al, 17          ; read error
000083B4 0F84AEF7FFFF        <1>       je     cd_drive_not_ready
                             <1> 
                             <1> loc_ln_file_dir_not_found:
000083BA BE[FE0F0100]        <1>       mov    esi, Msg_File_Directory_Not_Found
                             <1>       ;call  print_msg
                             <1>         ;retn
000083BF E999DFFFFF          <1>       jmp    print_msg
                             <1> 
                             <1> loc_longname_not_found:
000083C4 BE[1D100100]        <1>         mov     esi, Msg_LongName_Not_Found
                             <1>       ;call  print_msg
                             <1>         ;retn
000083C9 E98FDFFFFF          <1>       jmp    print_msg
                             <1> 
                             <1> loc_print_longname:
                             <1>       ;mov   esi, LongFileName
000083CE BF[FE590100]        <1>       mov    edi, TextBuffer
000083D3 57                  <1>       push   edi
000083D4 3C00                <1>       cmp    al, 0
000083D6 7708                <1>       ja     short loc_print_longname_1
                             <1> loc_print_FS_longname: ; Singlix FS (64 byte ASCIIZ file name)
000083D8 AC                  <1>       lodsb
000083D9 AA                  <1>       stosb
000083DA 08C0                <1>       or     al, al
000083DC 75FA                <1>       jnz    short loc_print_FS_longname
000083DE EB07                <1>       jmp    short loc_print_longname_2
                             <1>       ;
                             <1> loc_print_longname_1: ; MS Windows long name (UNICODE chars)
000083E0 66AD                <1>       lodsw
000083E2 AA                  <1>       stosb
000083E3 08C0                <1>       or     al, al
000083E5 75F9                <1>       jnz    short loc_print_longname_1
                             <1>       ;
                             <1> loc_print_longname_2:
000083E7 5E                  <1>       pop    esi
000083E8 E870DFFFFF          <1>       call   print_msg
000083ED BE[6F190100]        <1>       mov    esi, nextline
                             <1>       ;call  print_msg
                             <1>       ;retn
000083F2 E966DFFFFF          <1>       jmp    print_msg
                             <1> 
                             <1> show_file:
                             <1>       ; 18/02/2016
                             <1>       ; 17/02/2016
                             <1>       ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 13/09/2011 (CMD_INTR.ASM, 'cmp_cmd_show')
                             <1>       ; 08/11/2009
                             <1> 
                             <1> loc_show_parse_path_name:
000083F7 BF[A2620100]        <1>       mov    edi, FindFile_Drv
000083FC E833200000          <1>       call   parse_path_name
00008401 0F824CF9FFFF        <1>       jc     loc_cmd_failed
                             <1> 
                             <1> loc_show_check_filename_exists:
00008407 BE[E4620100]        <1>       mov    esi, FindFile_Name
0000840C 803E20              <1>       cmp    byte [esi], 20h
0000840F 0F863EF9FFFF        <1>       jna    loc_cmd_failed
                             <1> 
                             <1>       ; 15/02/2016 (invalid file name check)
00008415 E807020000          <1>       call   check_filename
0000841A 730A                <1>       jnc    short loc_show_change_drv
                             <1> 
0000841C BE[EA100100]        <1>       mov    esi, Msg_invalid_name_chars
00008421 E937DFFFFF          <1>       jmp    print_msg
                             <1> 
                             <1> loc_show_change_drv:
00008426 8A35[FE580100]      <1>       mov    dh, [Current_Drv]
0000842C 8835[5E610100]      <1>       mov    [RUN_CDRV], dh
00008432 8A15[A2620100]      <1>       mov    dl, [FindFile_Drv]
00008438 38F2                <1>       cmp    dl, dh
0000843A 740B                <1>       je     short loc_show_change_directory
0000843C E87FEAFFFF          <1>       call   change_current_drive
```

```
                               <1>       ;jc    loc_file_rw_cmd_failed
00008441 0F8237F9FFFF          <1>       jc     loc_run_cmd_failed
                               <1>
                               <1> loc_show_change_directory:
00008447 803D[A3620100]20      <1>       cmp    byte [FindFile_Directory], 20h
0000844E 7618                  <1>       jna    short loc_findload_showfile
                               <1>
00008450 FE05[D30C0100]        <1>       inc    byte [Restore_CDIR]
00008456 BE[A3620100]          <1>       mov    esi, FindFile_Directory
0000845B 30E4                  <1>       xor    ah, ah ; CD_COMMAND sign -> 0
0000845D E8BC190000            <1>       call   change_current_directory
                               <1>       ;jc    loc_file_rw_cmd_failed
00008462 0F8216F9FFFF          <1>       jc     loc_run_cmd_failed
                               <1>
                               <1> ;loc_show_change_prompt_dir_string:
                               <1>       ;call  change_prompt_dir_string
                               <1>
                               <1> loc_findload_showfile:
                               <1>       ; 15/02/2016
00008468 BE[E4620100]          <1>       mov    esi, FindFile_Name
0000846D BF[93620100]          <1>       mov    edi, Dir_Entry_Name ; Dir Entry Format File Name
00008472 E8F21E0000            <1>       call   convert_file_name
00008477 89FE                  <1>       mov    esi, edi ; offset Dir_Entry_Name
                               <1>
00008479 28C0                  <1>       sub    al, al ; Attrib AND mask = 0
                               <1>       ; Directory attribute : 10h
                               <1>       ; Volume name attribute: 8h
0000847B B418                  <1>       mov    ah, 00011000b ; 18h (Attrib NAND, AND --> zero mask)
                               <1>       ;
0000847D 6631C9                <1>       xor    cx, cx
00008480 E8ED1B0000            <1>       call   locate_current_dir_file
                               <1>       ;jc    loc_file_rw_cmd_failed
00008485 0F82F3F8FFFF          <1>       jc     loc_run_cmd_failed
                               <1>
                               <1> loc_show_load_file:
                               <1>       ; EDI = Directory Entry
0000848B 668B4714              <1>       mov    ax, [edi+DirEntry_FstClusHI] ; First Cluster High Word
0000848F C1E010                <1>       shl    eax, 16
00008492 668B471A              <1>       mov    ax, [edi+DirEntry_FstClusLO] ; First Cluster Low Word
00008496 A3[4C630100]          <1>       mov    [Show_Cluster], eax
0000849B 8B471C                <1>       mov    eax, [edi+DirEntry_FileSize] ; File Size
0000849E 21C0                  <1>       and    eax, eax ; Empty file !
000084A0 0F8491000000          <1>         jz      end_of_show_file
000084A6 A3[50630100]          <1>       mov    [Show_FileSize], eax
000084AB 31C0                  <1>       xor    eax, eax
000084AD A3[54630100]          <1>       mov    [Show_FilePointer], eax ; 0
000084B2 66A3[58630100]        <1>       mov    [Show_ClusterPointer], ax ; 0
000084B8 29DB                  <1>       sub    ebx, ebx
000084BA 8A3D[FE580100]        <1>       mov    bh, [Current_Drv]
000084C0 BE00010900            <1>       mov    esi, Logical_DOSDisks
000084C5 01DE                  <1>       add    esi, ebx
000084C7 8935[48630100]        <1>       mov    [Show_LDDDT], esi ; Logical DOS Drv Description Table addr
                               <1>
000084CD 807E0300              <1>       cmp    byte [esi+LD_FATType], 0
000084D1 7713                  <1>       ja     short loc_show_calculate_cluster_size
                               <1>       ; Singlix FS
                               <1>       ; First Cluster Number is FDT number (in compatibility buffer)
000084D3 8B15[4C630100]        <1>       mov    edx, [Show_Cluster] ; Compatibility dir. buffer value (FDT)
000084D9 8915[44630100]        <1>       mov    [Show_FDT], edx
000084DF 31C0                  <1>       xor    eax, eax
000084E1 A3[4C630100]          <1>       mov    [Show_Cluster], eax ; Sector index  = 0
                               <1>                                 ; (next time it will be 1)
                               <1> loc_show_calculate_cluster_size:
000084E6 668B5E11              <1>       mov    bx, [esi+LD_BPB+BPB_BytsPerSec] ; FAT 12-16-32 (512)
                               <1>       ; BX = 512 = [esi+LD_FS_BytesPerSec] ; Singlix FS
000084EA 8A4613                <1>       mov    al, [esi+LD_BPB+BPB_SecPerClust] ; FAT 12-16-32 (<= 128)
                               <1>       ; AL = 1 = [esi+LD_FS_Reserved2] ; SectPerClust for Singlix FS
000084ED F7E3                  <1>       mul    ebx
                               <1>
                               <1>       ;cmp   eax, 65536 ; non-compatible (very big) cluster size
                               <1>       ;ja    short end_of_show_file
000084EF 66A3[5A630100]        <1>       mov    [Show_ClusterSize], ax
                               <1>
                               <1> loc_start_show_file:
000084F5 BE[6F190100]          <1>       mov    esi, nextline
000084FA E85EDEFFFF            <1>       call   print_msg
                               <1>
000084FF A1[4C630100]          <1>       mov    eax, [Show_Cluster]
00008504 C605[5C630100]17      <1>       mov    byte [Show_RowCount], 23
                               <1>
                               <1>       ; 17/02/2016
0000850B 8B35[48630100]        <1>       mov    esi, [Show_LDDDT]
                               <1>
                               <1> loc_show_next_cluster:
                               <1>       ; 15/02/2016
00008511 BB00000700            <1>       mov    ebx, Cluster_Buffer ; 70000h (for current TRDOS 386 version)
                               <1>       ; ESI = Logical DOS drv description table address
00008516 E851380000            <1>       call   read_cluster
                               <1>       ;jc    loc_file_rw_cmd_failed
0000851B 0F825DF8FFFF          <1>       jc     loc_run_cmd_failed
                               <1>
00008521 31DB                  <1>       xor    ebx, ebx
                               <1> loc_show_next_byte:
00008523 803D[5C630100]00      <1>       cmp    byte [Show_RowCount], 0
0000852A 7521                  <1>       jne    short pass_show_wait_for_key
0000852C 30E4                  <1>       xor    ah, ah
0000852E E8E386FFFF            <1>       call   int16h
00008533 3C1B                  <1>       cmp    al, 1Bh
00008535 750F                  <1>       jne    short pass_exit_show
                               <1> end_of_show_file:
                               <1> pass_show_file:
00008537 BE[6F190100]          <1>       mov    esi, nextline
0000853C E81CDEFFFF            <1>       call   print_msg
```

```
00008541 E94B010000           <1>       jmp    loc_file_rw_restore_retn
                              <1> 
                              <1> pass_exit_show:
00008546 C605[5C630100]14     <1>       mov    byte [Show_RowCount], 20
                              <1> pass_show_wait_for_key:
0000854D 81C300000700         <1>       add    ebx, Cluster_Buffer
00008553 8A03                 <1>       mov    al, [ebx]
00008555 3C0D                 <1>       cmp    al, 0Dh
00008557 0F8590000000         <1>         jne     loc_show_check_tab_space
0000855D FE0D[5C630100]       <1>       dec    byte [Show_RowCount]
                              <1> pass_show_dec_rowcount:
00008563 B307                 <1>       mov    bl, 7 ; (light gray character color, black background)
00008565 8A3D[66580100]       <1>       mov    bh, [ACTIVE_PAGE] ; [ptty]
0000856B E84297FFFF           <1>       call   _write_tty
                              <1> loc_show_check_eof:
00008570 FF05[54630100]       <1>       inc    dword [Show_FilePointer]
00008576 A1[54630100]         <1>       mov    eax, [Show_FilePointer]
0000857B 3B05[50630100]       <1>       cmp    eax, [Show_FileSize]
00008581 73B4                 <1>       jnb    short end_of_show_file
00008583 66FF05[58630100]     <1>       inc    word [Show_ClusterPointer]
0000858A 0FB71D[58630100]     <1>       movzx  ebx, word [Show_ClusterPointer]
                              <1> 
                              <1>       ; 17/02/2016
                              <1>       ; (sector boundary -9 bits- check, 512 = 0)
00008591 66F7C3FF01           <1>         test    bx, 1FFh ;  1 to 511
00008596 758B                 <1>       jnz    short loc_show_next_byte
                              <1> 
                              <1>       ; 16/02/2016
00008598 8B35[48630100]       <1>       mov    esi, [Show_LDDDT]
                              <1>       ;
0000859E 807E0300             <1>       cmp    byte [esi+LD_FATType], 0
000085A2 7719                 <1>       ja     short loc_show_check_fat_cluster_size
                              <1> 
                              <1>       ; Singlix FS
                              <1>       ; 1 sector, more... (cluster size = 1 sector)
000085A4 A1[4C630100]         <1>       mov    eax, [Show_Cluster]
000085A9 40                   <1>       inc    eax
000085AA A3[4C630100]         <1>       mov    [Show_Cluster], eax
                              <1> 
000085AF 6621DB               <1>       and    bx, bx ; 65536 -> 0
000085B2 0F856BFFFFFF         <1>         jnz  loc_show_next_byte
000085B8 E954FFFFFF           <1>       jmp      loc_show_next_cluster
                              <1> 
                              <1> loc_show_check_fat_cluster_size:
                              <1>       ; 17/02/2016
000085BD 663B1D[5A630100]     <1>       cmp    bx, [Show_ClusterSize] ; cluster size in bytes
000085C4 0F8259FFFFFF         <1>         jb   loc_show_next_byte
000085CA 66C705[58630100]00- <1>       mov    word [Show_ClusterPointer], 0
000085D2 00                   <1>
                              <1> 
000085D3 A1[4C630100]         <1>       mov    eax, [Show_Cluster]
                              <1>       ;mov   esi, [Show_LDDDT]
                              <1> loc_show_get_next_cluster:
000085D8 E86B350000           <1>       call   get_next_cluster
                              <1>       ;jc    loc_file_rw_cmd_failed
000085DD 0F829BF7FFFF         <1>       jc     loc_run_cmd_failed
                              <1> loc_show_update_ccluster:
000085E3 A3[4C630100]         <1>       mov    [Show_Cluster], eax
000085E8 E924FFFFFF           <1>         jmp     loc_show_next_cluster
                              <1> 
                              <1> loc_show_check_tab_space:
000085ED 3C09                 <1>       cmp    al, 09h
000085EF 0F856EFFFFFF         <1>         jne     pass_show_dec_rowcount
                              <1> loc_show_put_tab_space:
000085F5 8A3D[66580100]       <1>       mov    bh, [ACTIVE_PAGE] ; [ptty]
000085FB E84193FFFF           <1>       call   get_cpos
                              <1>       ; dl = cursor column
00008600 80E207               <1>       and    dl, 7 ; 18/02/2016
                              <1>       ;shr   bh, 1 ; [ACTIVE_PAGE]
00008603 8A3D[66580100]       <1>       mov    bh, [ACTIVE_PAGE]
00008609 B307                 <1>       mov    bl, 7 ; color attribute
                              <1> loc_show_put_space_chars:
0000860B B020                 <1>       mov    al, 20h ; space
                              <1>       ;mov   bh, [ACTIVE_PAGE] ; [ptty]
                              <1>       ;mov   bl, 7 ; color attribute
                              <1>       ;push  dx
0000860D 52                   <1>       push   edx ; 29/12/2017
0000860E E89F96FFFF           <1>       call   _write_tty
00008613 5A                   <1>       pop    edx ; 29/12/2017
                              <1>       ;pop   dx
                              <1>       ; 18/02/2016
00008614 80FA07               <1>       cmp    dl, 7
00008617 0F8353FFFFFF         <1>       jnb    loc_show_check_eof
0000861D FEC2                 <1>       inc    dl
0000861F EBEA                 <1>       jmp    short loc_show_put_space_chars
                              <1> 
                              <1> check_filename:
                              <1>       ; 10/10/2016
                              <1>       ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
                              <1>       ; 07/08/2010 (FILE.ASM, 'proc_check_filename')
                              <1>       ; 10/07/2010
                              <1>       ; Derived from 'proc_check_filename'
                              <1>       ; in the old TRDOS.ASM (09/02/2005).
                              <1>       ;
                              <1>       ; INPUT ->
                              <1>       ;      ESI = Dot File Name Location
                              <1>       ; OUTPUT ->
                              <1>       ;      cf = 1 -> error code in AL
                              <1>       ;           AL = ERR_INV_FILE_NAME (=26)
                              <1>       ;              Invalid file name chars
                              <1>       ;      cf = 0 -> valid file name
                              <1>       ;
                              <1>       ;(EAX, ECX, EDI will be changed)
```

```
                             <1> 
                             <1> check_invalid_filename_chars:
                             <1>       ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 10/07/2010 (FILE.ASM, 'proc_check_invalid_filename_chars')
                             <1>       ; 10/02/2010
                             <1>       ; Derived from 'proc_check_invalid_filename_chars'
                             <1>       ; in the old TRDOS.ASM (09/02/2005).
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      ESI = ASCIIZ FileName
                             <1>       ; OUTPUT ->
                             <1>       ;      cf = 1 -> invalid
                             <1>       ;      cf = 0 -> valid
                             <1>       ;
                             <1>       ;(EAX, ECX, EDI will be changed)
                             <1> 
00008621 56                  <1>       push   esi
                             <1> 
00008622 BF[D20D0100]        <1>         mov     edi, invalid_fname_chars
00008627 AC                  <1>       lodsb
                             <1> check_filename_next_char:
00008628 B914000000          <1>       mov    ecx, sizeInvFnChars
0000862D BF[D20D0100]        <1>       mov    edi, invalid_fname_chars
                             <1> loc_scan_invalid_filename_char:
00008632 AE                  <1>       scasb
00008633 741F                <1>       je     short loc_invalid_filename_stc
00008635 E2FB                <1>       loop   loc_scan_invalid_filename_char
00008637 AC                  <1>       lodsb
00008638 3C1F                <1>       cmp    al, 1Fh  ; 20h and above
0000863A 77EC                <1>       ja     short check_filename_next_char
                             <1> 
                             <1> check_filename_dot:
0000863C 8B3424              <1>       mov    esi, [esp]
                             <1> 
0000863F B421                <1>       mov    ah, 21h
00008641 B908000000          <1>       mov    ecx, 8
                             <1> loc_check_filename_next_char:
00008646 AC                  <1>       lodsb
00008647 3C2E                <1>       cmp    al, 2Eh
00008649 7511                <1>       jne    short pass_check_fn_dot_check
                             <1> loc_check_filename_ext_0:
0000864B AC                  <1>       lodsb
0000864C 38E0                <1>       cmp    al, ah ; 21h
0000864E 7205                <1>       jb     short loc_invalid_filename
00008650 3C2E                <1>       cmp    al, 2Eh
00008652 7519                <1>       jne    short loc_check_filename_ext_1
                             <1> 
                             <1> loc_invalid_filename_stc:
                             <1> loc_check_fn_stc_rtn:
00008654 F9                  <1>       stc
                             <1> loc_invalid_filename:
                             <1>       ; 10/10/2016 (0Bh -> 26)
00008655 B81A000000          <1>       mov    eax, ERR_INV_FILE_NAME ; (=26)
                             <1>       ; Invalid file name chars
                             <1> loc_check_fn_rtn:
0000865A 5E                  <1>       pop    esi
0000865B C3                  <1>       retn
                             <1> 
                             <1> pass_check_fn_dot_check:
0000865C 38E0                <1>       cmp    al, ah ; 21h
0000865E 7224                <1>       jb     short loc_check_fn_clc_rtn
00008660 E2E4                <1>       loop   loc_check_filename_next_char
00008662 AC                  <1>       lodsb
00008663 38E0                <1>       cmp    al, ah ; 21h
00008665 721D                <1>       jb     short loc_check_fn_clc_rtn
00008667 3C2E                <1>       cmp    al, 2Eh
00008669 75E9                <1>       jne    short loc_check_fn_stc_rtn
0000866B EBDE                <1>       jmp    short loc_check_filename_ext_0
                             <1> 
                             <1> loc_check_filename_ext_1:
0000866D AC                  <1>       lodsb
0000866E 38E0                <1>       cmp    al, ah ; 21h
00008670 7212                <1>       jb     short loc_check_fn_clc_rtn
00008672 3C2E                <1>       cmp    al, 2Eh
00008674 74DE                <1>       je     short loc_check_fn_stc_rtn
00008676 AC                  <1>       lodsb
00008677 38E0                <1>       cmp    al, ah ; 21h
00008679 7209                <1>       jb     short loc_check_fn_clc_rtn
0000867B 3C2E                <1>       cmp    al, 2Eh
0000867D 74D5                <1>       je     short loc_check_fn_stc_rtn
0000867F AC                  <1>       lodsb
00008680 38E0                <1>       cmp    al, ah ; 21h
00008682 73D0                <1>       jnb    short loc_check_fn_stc_rtn
                             <1> 
                             <1> loc_check_fn_clc_rtn:
00008684 5E                  <1>       pop    esi
00008685 F8                  <1>       clc
00008686 C3                  <1>       retn
                             <1> 
                             <1> loc_print_deleted_message:
00008687 BE[BF110100]        <1>       mov    esi, Msg_Deleted
0000868C E8CCDCFFFF          <1>       call   print_msg
                             <1> 
                             <1>       ;clc
                             <1> 
                             <1> loc_file_rw_restore_retn:
                             <1>       ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 28/02/2010 (CMD_INTR.ASM)
                             <1> loc_file_rw_cmd_failed:
00008691 9C                  <1>       pushf
00008692 E84FF7FFFF          <1>       call   restore_cdir_after_cmd_fail
00008697 9D                  <1>       popf
00008698 720D                <1>       jc     short loc_file_rw_check_write_fault
```

```
0000869A C3                  <1>       retn
                             <1>
                             <1> loc_permission_denied:
                             <1>       ; 27/02/2016
0000869B BE[CC110100]        <1>       mov    esi, Msg_Permission_Denied
000086A0 E8B8DCFFFF          <1>       call   print_msg
000086A5 EBEA                <1>       jmp    short loc_file_rw_restore_retn
                             <1>
                             <1> loc_file_rw_check_write_fault:
                             <1>       ;cmp   al, 1Dh ; Write Fault
000086A7 3C12                <1>         cmp  al, 18 ; 05/11/2016
000086A9 0F85D4F6FFFF        <1>       jne     loc_run_cmd_failed_cmp_al
000086AF BE[B30F0100]        <1>       mov    esi, Msg_Not_Ready_Write_Err
                             <1>       ;call  print_msg
                             <1>       ;retn
000086B4 E9A4DCFFFF          <1>       jmp    print_msg
                             <1>
                             <1> make_directory:
                             <1>       ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 12/03/2011 (CMD_INTR.ASM, 'cmp_cmd_mkdir')
                             <1>       ; 14/08/2010
                             <1>       ; 10/07/2010
                             <1>       ; 29/11/2009
                             <1>       ;
                             <1> get_mkdir_fchar:
                             <1>       ; esi = directory name
000086B9 803E20              <1>       cmp    byte [esi], 20h
000086BC 7701                <1>         ja   short loc_mkdir_parse_path_name
                             <1>
                             <1> loc_mkdir_nodirname_retn:
000086BE C3                  <1>       retn
                             <1>
                             <1> loc_mkdir_parse_path_name:
000086BF BF[A2620100]        <1>       mov    edi, FindFile_Drv
000086C4 E86B1D0000          <1>         call    parse_path_name
000086C9 0F8284F6FFFF        <1>       jc     loc_cmd_failed
                             <1>
                             <1> loc_mkdir_check_dirname_exists:
000086CF BE[E4620100]        <1>       mov    esi, FindFile_Name
000086D4 803E20              <1>       cmp    byte [esi], 20h
000086D7 0F8676F6FFFF        <1>       jna    loc_cmd_failed
000086DD 8935[60630100]      <1>       mov    [DelFile_FNPointer], esi
000086E3 E839FFFFFF          <1>       call   check_filename
000086E8 7259                <1>       jc     short loc_mkdir_invalid_dir_name_chars
                             <1>
                             <1> loc_mkdir_drv:
000086EA 8A35[FE580100]      <1>       mov    dh, [Current_Drv]
000086F0 8835[5E610100]      <1>       mov    [RUN_CDRV], dh
                             <1>
000086F6 8A15[A2620100]      <1>       mov    dl, [FindFile_Drv]
000086FC 38F2                <1>       cmp    dl, dh
000086FE 7407                <1>       je     short loc_mkdir_change_directory
                             <1>
00008700 E8BBE7FFFF          <1>       call   change_current_drive
00008705 728A                <1>       jc     loc_file_rw_cmd_failed
                             <1>
                             <1> loc_mkdir_change_directory:
00008707 803D[A3620100]20    <1>       cmp    byte [FindFile_Directory], 20h
0000870E 7614                <1>       jna    short loc_mkdir_find_directory
                             <1>
00008710 FE05[D30C0100]      <1>       inc    byte [Restore_CDIR]
00008716 BE[A3620100]        <1>       mov    esi, FindFile_Directory
0000871B 30E4                <1>       xor    ah, ah ; CD_COMMAND sign -> 0
0000871D E8FC160000          <1>       call   change_current_directory
00008722 722E                <1>       jc     short loc_mkdir_check_error_code
                             <1>
                             <1> ;loc_mkdir_change_prompt_dir_string:
                             <1>       ;call  change_prompt_dir_string
                             <1>
                             <1> loc_mkdir_find_directory:
                             <1>       ;mov   esi, FindFile_Name
00008724 8B35[60630100]      <1>       mov    esi, [DelFile_FNPointer]
                             <1>       ;xor   eax, eax
0000872A 6631C0              <1>       xor    ax, ax ; any name (dir, file, volume)
0000872D E831FBFFFF          <1>       call   find_first_file
00008732 721E                <1>       jc     short loc_mkdir_check_error_code
                             <1>
                             <1> loc_mkdir_directory_found:
00008734 BE[17110100]        <1>       mov    esi, Msg_Name_Exists
00008739 E81FDCFFFF          <1>       call   print_msg
                             <1>
0000873E E94EFFFFFF          <1>         jmp     loc_file_rw_restore_retn
                             <1>
                             <1> loc_mkdir_invalid_dir_name_chars:
00008743 BE[EA100100]        <1>       mov    esi, Msg_invalid_name_chars
00008748 E810DCFFFF          <1>       call   print_msg
                             <1>
0000874D E93FFFFFFF          <1>         jmp     loc_file_rw_restore_retn
                             <1>
                             <1> loc_mkdir_check_error_code:
00008752 3C02                <1>       cmp    al, 2
                             <1>       ;je    short loc_mkdir_directory_not_found
00008754 7406                <1>       je     short loc_mkdir_ask_for_yes_no
00008756 F9                  <1>       stc
00008757 E935FFFFFF          <1>         jmp     loc_file_rw_cmd_failed
                             <1>
                             <1> loc_mkdir_directory_not_found:
                             <1> loc_mkdir_ask_for_yes_no:
0000875C BE[38110100]        <1>       mov    esi, Msg_DoYouWantMkdir
00008761 E8F7DBFFFF          <1>       call   print_msg
00008766 8B35[60630100]      <1>       mov    esi, [DelFile_FNPointer]
0000876C E8ECDBFFFF          <1>       call   print_msg
00008771 BE[57110100]        <1>       mov    esi, Msg_YesNo
```

```
00008776 E8E2DBFFFF           <1>       call   print_msg
                              <1> 
0000877B C605[61110100]20     <1>       mov    byte [Y_N_nextline], 20h
                              <1> 
                              <1> loc_mkdir_ask_again:
00008782 30E4                 <1>       xor    ah, ah
00008784 E88D84FFFF           <1>       call   int16h
00008789 3C1B                 <1>       cmp    al, 1Bh
                              <1>       ;je    short loc_do_not_make_directory
0000878B 7439                 <1>       je     short loc_mkdir_y_n_escape
0000878D 24DF                 <1>       and    al, 0DFh ; y -> Y, n -> N
0000878F 3C59                 <1>       cmp    al, 'Y' ; 'yes'
00008791 7404                 <1>       je     short loc_mkdir_yes_make_directory
00008793 3C4E                 <1>       cmp    al, 'N' ; 'no'
00008795 75EB                 <1>       jne    short loc_mkdir_ask_again
                              <1> 
                              <1> loc_do_not_make_directory:
                              <1> loc_mkdir_yes_make_directory:
00008797 E82E000000           <1>       call   y_n_answer ; 29/12/2017
                              <1>       ;cmp   al, 'Y' ; 'yes'
                              <1>       ;cmc
                              <1>         ;jnc loc_file_rw_restore_retn
0000879C 3C4E                 <1>       cmp    al, 'N' ; 'no'
0000879E 0F84EDFEFFFF         <1>         je   loc_file_rw_restore_retn
                              <1> 
                              <1> loc_mkdir_call_make_sub_directory:
000087A4 8B35[60630100]       <1>       mov    esi, [DelFile_FNPointer]
000087AA B110                 <1>       mov    cl, 10h ; Directory attributes
000087AC E8821D0000           <1>       call   make_sub_directory
                              <1> loc_rename_file_ok: ; 06/03/2016
000087B1 0F82DAFEFFFF         <1>         jc   loc_file_rw_cmd_failed
                              <1> move_source_file_to_destination_OK:
000087B7 BE[65110100]         <1>       mov    esi, Msg_OK
000087BC E89CDBFFFF           <1>       call   print_msg
000087C1 E9CBFEFFFF           <1>       jmp    loc_file_rw_restore_retn
                              <1> 
                              <1> loc_mkdir_y_n_escape:
000087C6 B04E                 <1>       mov    al, 'N' ; 'no'
000087C8 EBCD                 <1>       jmp    short loc_do_not_make_directory
                              <1> 
                              <1> y_n_answer:
                              <1>       ; 29/12/2017
000087CA A2[61110100]         <1>       mov    [Y_N_nextline], al
                              <1>       ;push  ax
000087CF 50                   <1>       push   eax
000087D0 BE[61110100]         <1>       mov    esi, Y_N_nextline
000087D5 E883DBFFFF           <1>       call   print_msg
000087DA 58                   <1>       pop    eax
                              <1>       ;pop   ax
000087DB C3                   <1>       retn
                              <1> 
                              <1> delete_directory:
                              <1>       ; 29/12/2017
                              <1>       ; 15/10/2016
                              <1>       ; 01/03/2016, 06/03/2016
                              <1>       ; 27/02/2016, 28/02/2016, 29/02/2016
                              <1>       ; 26/02/2016 (TRDOS 386 = TRDOS v2.0)
                              <1>       ; 16/10/2010 (CMD_INTR.ASM, 'cmp_cmd_rmdir')
                              <1>       ; 05/06/2010
                              <1>       ;
                              <1> get_fchar:
                              <1>       ; esi = directory name
000087DC 803E20               <1>       cmp    byte [esi], 20h
000087DF 7701                 <1>         ja   short loc_rmdir_parse_path_name
                              <1> 
                              <1> loc_rmdir_nodirname_retn:
000087E1 C3                   <1>       retn
                              <1> 
                              <1> loc_rmdir_parse_path_name:
000087E2 BF[A2620100]         <1>       mov    edi, FindFile_Drv
000087E7 E8481C0000           <1>       call   parse_path_name
000087EC 0F8261F5FFFF         <1>       jc     loc_cmd_failed
                              <1> 
                              <1> loc_rmdir_check_dirname_exists:
000087F2 BE[E4620100]         <1>       mov    esi, FindFile_Name
000087F7 803E20               <1>       cmp    byte [esi], 20h
000087FA 0F8653F5FFFF         <1>       jna    loc_cmd_failed
00008800 8935[60630100]       <1>       mov    [DelFile_FNPointer], esi
                              <1> 
                              <1> loc_rmdir_drv:
00008806 8A35[FE580100]       <1>       mov    dh, [Current_Drv]
0000880C 8835[5E610100]       <1>       mov    [RUN_CDRV], dh
                              <1> 
00008812 8A15[A2620100]       <1>       mov    dl, [FindFile_Drv]
00008818 38F2                 <1>       cmp    dl, dh
0000881A 740B                 <1>       je     short loc_rmdir_change_directory
                              <1> 
0000881C E89FE6FFFF           <1>       call   change_current_drive
00008821 0F826AFEFFFF         <1>       jc     loc_file_rw_cmd_failed
                              <1> 
                              <1> loc_rmdir_change_directory:
00008827 803D[A3620100]20     <1>       cmp    byte [FindFile_Directory], 20h
0000882E 7614                 <1>       jna    short loc_rmdir_find_directory
                              <1> 
00008830 FE05[D30C0100]       <1>       inc    byte [Restore_CDIR]
00008836 BE[A3620100]         <1>       mov    esi, FindFile_Directory
0000883B 30E4                 <1>       xor    ah, ah ; CD_COMMAND sign -> 0
0000883D E8DC150000           <1>       call   change_current_directory
00008842 7211                 <1>       jc     short loc_rmdir_check_error_code
                              <1> 
                              <1> ;loc_rmdir_change_prompt_dir_string:
                              <1>       ;call  change_prompt_dir_string
                              <1> 
```

```
                             <1> loc_rmdir_find_directory:
                             <1>       ;mov   esi, FindFile_Name
00008844 8B35[60630100]      <1>       mov    esi, [DelFile_FNPointer]
0000884A 66B81008            <1>       mov    ax, 0810h ; Only directories
0000884E E810FAFFFF          <1>       call   find_first_file
00008853 730A                <1>       jnc    short loc_rmdir_ambgfn_check
                             <1>
                             <1> loc_rmdir_check_error_code:
00008855 3C02                <1>       cmp    al, 2
00008857 740B                <1>       je     short loc_rmdir_directory_not_found
00008859 F9                  <1>       stc
0000885A E932FEFFFF          <1>       jmp    loc_file_rw_cmd_failed
                             <1>
                             <1> loc_rmdir_ambgfn_check:
0000885F 6621D2              <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
00008862 740F                <1>       jz     short loc_rmdir_directory_found
                             <1>
                             <1> loc_rmdir_directory_not_found:
00008864 BE[D50F0100]        <1>       mov    esi, Msg_Dir_Not_Found
00008869 E8EFDAFFFF          <1>       call   print_msg
                             <1>
0000886E E91EFEFFFF          <1>       jmp    loc_file_rw_restore_retn
                             <1>
                             <1> loc_rmdir_directory_found:
00008873 80E307              <1>       and    bl, 07h ; Attributes
00008876 0F851FFEFFFF        <1>       jnz    loc_permission_denied
                             <1>
                             <1> loc_rmdir_save_lnel: ; 28/02/2016
                             <1>        ;mov  bh, [LongName_EntryLength]
0000887C 883D[6A630100]      <1>       mov    [DelFile_LNEL], bh ; Long name entry length (if > 0)
                             <1>       ; edi = Directory Entry Offset (DirBuff)
                             <1>       ; esi = Directory Entry (FFF Structure)
                             <1>       ;mov   [DelFile_DirEntryAddr], edi ; not required
                             <1>       ;mov   ax, [edi+20] ; First Cluster High Word
                             <1>         ;shl eax, 16
                             <1>       ;mov   ax, [edi+26] ; First Cluster Low Word
                             <1>       ; ROOT Dir First Cluster = 0
                             <1>         ;cmp eax, 2
                             <1>       ;jb    loc_update_direntry_1
                             <1>
                             <1> pass_rmdir_fc_check:
00008882 57                  <1>       push   edi ; * (29/02/2016)
                             <1>
00008883 BE[6B110100]        <1>       mov    esi, Msg_DoYouWantRmDir
00008888 E8D0DAFFFF          <1>       call   print_msg
0000888D 8B35[60630100]      <1>       mov    esi, [DelFile_FNPointer]
00008893 E8C5DAFFFF          <1>       call   print_msg
00008898 BE[57110100]        <1>       mov    esi, Msg_YesNo
0000889D E8BBDAFFFF          <1>       call   print_msg
                             <1>
                             <1> loc_rmdir_ask_again:
000088A2 30E4                <1>       xor    ah, ah
000088A4 E86D83FFFF          <1>       call   int16h
000088A9 3C1B                <1>       cmp    al, 1Bh
                             <1>       ;je    short loc_do_not_delete_directory
000088AB 7433                <1>         je      loc_rmdir_y_n_escape ; 06/03/2016
000088AD 24DF                <1>       and    al, 0DFh
000088AF A2[61110100]        <1>       mov    [Y_N_nextline], al
000088B4 3C59                <1>       cmp    al, 'Y'
000088B6 7404                <1>       je     short loc_rmdir_yes_delete_directory
000088B8 3C4E                <1>       cmp    al, 'N'
000088BA 75E6                <1>       jne    short loc_rmdir_ask_again
                             <1>
                             <1> loc_do_not_delete_directory:
                             <1> loc_rmdir_yes_delete_directory:
000088BC E809FFFFFF          <1>       call   y_n_answer ; 29/12/2017
000088C1 5F                  <1>       pop    edi ; * (29/02/2016)
                             <1>       ;cmp   al, 'Y' ; 'yes'
                             <1>       ;cmc
                             <1>         ;jnc loc_file_rw_restore_retn
000088C2 3C4E                <1>       cmp    al, 'N' ; 'no'
000088C4 0F84C7FDFFFF        <1>         je   loc_file_rw_restore_retn
                             <1>
                             <1>       ; 29/12/2017
000088CA E869000000          <1>       call   delete_sub_directory
000088CF 7213                <1>       jc     short loc_rmdir_cmd_failed
                             <1>
                             <1> loc_rmdir_ok:
000088D1 BE[65110100]        <1>       mov    esi, Msg_OK
000088D6 E882DAFFFF          <1>       call   print_msg
000088DB E9B1FDFFFF          <1>       jmp    loc_file_rw_restore_retn
                             <1>
                             <1> loc_rmdir_y_n_escape:
000088E0 B04E                <1>       mov    al, 'N' ; 'no'
000088E2 EBD8                <1>         jmp     loc_do_not_delete_directory
                             <1>
                             <1> loc_rmdir_cmd_failed:
                             <1>       ; 29/12/2017
000088E4 09C0                <1>       or     eax, eax ; EAX = 0 -> Directory not empty!
000088E6 7426                <1>       jz     short loc_rmdir_directory_not_empty
                             <1>
                             <1>       ; EAX > 0 -> Error code in AL (or AX or EAX)
                             <1>
000088E8 833D[1E610100]01    <1>       cmp    dword [FAT_ClusterCounter], 1
000088EF 0F829CFDFFFF        <1>       jb     loc_file_rw_cmd_failed
000088F5 F9                  <1>       stc
                             <1> loc_rmdir_cmd_return:
                             <1>       ; 01/03/2016
000088F6 9C                  <1>       pushf
                             <1>       ; ESI = Logical DOS Drive Description Table address
000088F7 66BB00FF            <1>       mov    bx, 0FF00h ; BH = FFh -> use ESI for Drive parameters
                             <1>                  ; BL = 0 -> Recalculate free cluster count
000088FB 50                  <1>       push   eax
```

```
000088FC E8C3380000           <1>       call   calculate_fat_freespace
00008901 58                   <1>       pop    eax
00008902 9D                   <1>       popf
00008903 0F8288FDFFFF         <1>       jc     loc_file_rw_cmd_failed
00008909 E983FDFFFF           <1>       jmp    loc_file_rw_restore_retn
                              <1>
                              <1> loc_rmdir_directory_not_empty:
0000890E BE[8C110100]         <1>       mov    esi, Msg_Dir_Not_Empty
00008913 E845DAFFFF           <1>       call   print_msg
                              <1>       ; 01/03/2016
00008918 A1[1E610100]         <1>       mov    eax, [FAT_ClusterCounter]
0000891D 09C0                 <1>       or     eax, eax ; 0 ?
0000891F 0F846CFDFFFF         <1>       jz     loc_file_rw_restore_retn
                              <1>       ; ESI = Logical DOS Drive Description Table address
00008925 66BB01FF             <1>       mov    bx, 0FF01h ; BH = FFh -> use ESI for Drive parameters
                              <1>                   ; BL = 1 -> add free clusters
00008929 E896380000           <1>       call   calculate_fat_freespace
0000892E 09C9                 <1>       or     ecx, ecx
00008930 0F845BFDFFFF         <1>         jz      loc_file_rw_restore_retn ; ecx = 0 -> OK
                              <1>       ; ecx > 0 -> Error (Recalculation is needed)
00008936 EBBE                 <1>       jmp    short loc_rmdir_cmd_return
                              <1>
                              <1>
                              <1> delete_sub_directory:
                              <1>       ; 29/12/2017
                              <1>       ; (moved here from 'delete_directory' for 'sysrmdir' )
                              <1>
                              <1>       ; EDI = Directory buffer entry offset/address
                              <1>
                              <1> loc_rmdir_delete_short_name_check_dir_empty:
00008938 668B4714             <1>       mov    ax, [edi+20] ; First Cluster High Word
0000893C C1E010               <1>         shl  eax, 16
0000893F 668B471A             <1>       mov    ax, [edi+26] ; First Cluster Low Word
                              <1>
                              <1>       ;mov   [DelFile_FCluster], eax
                              <1>
                              <1>       ;;mov  bx, [DirBuff_EntryCounter]
                              <1>       ;mov   bx, [FindFile_DirEntryNumber] ; 27/02/2016
                              <1>       ;mov   [DelFile_EntryCounter], bx
                              <1>
00008943 29DB                 <1>       sub    ebx, ebx
                              <1>       ; 29/12/2017
00008945 891D[1E610100]       <1>       mov    [FAT_ClusterCounter], ebx ; 0 ; Reset
                              <1>
0000894B 8A3D[A2620100]       <1>       mov    bh, [FindFile_Drv]
00008951 BE00010900           <1>       mov    esi, Logical_DOSDisks
00008956 01DE                 <1>       add    esi, ebx
                              <1>
00008958 66817F0CA101         <1>       cmp    word [edi+DirEntry_NTRes], 01A1h
0000895E 745A                 <1>       je     short loc_rmdir_check_fs_directory
                              <1>
                              <1>       ;cmp   byte [esi+LD_FATType], 1
                              <1>       ;jb    short loc_rmdir_get__last_cluster_0
                              <1>
                              <1>       ; 29/12/2017
00008960 83F802               <1>       cmp    eax, 2
00008963 7306                 <1>       jnb    short loc_rmdir_get_last_cluster_1
                              <1>       ; eax < 2
                              <1> loc_rmdir_get_last_cluster_0:
                              <1>       ;mov   eax, ERR_INV_FORMAT ; invalid format!
00008965 B813000000           <1>       mov    eax, ERR_NOT_DIR ; not a valid directory!
                              <1>       ;stc
0000896A C3                   <1>       retn
                              <1>
                              <1> loc_rmdir_get_last_cluster_1:
0000896B 807E0303             <1>       cmp    byte [esi+LD_FATType], 3 ; FAT32
0000896F 750C                 <1>       jne    short loc_rmdir_get_last_cluster_2
                              <1>
                              <1>       ; is it root directory ?
00008971 3B4632               <1>       cmp    eax, [esi+LD_BPB+BPB_RootClus]
00008974 7507                 <1>       jne    short loc_rmdir_get_last_cluster_2
                              <1>
                              <1>       ; root directory can not be deleted !!
                              <1> loc_rmdir_permission_denied:
00008976 B80B000000           <1>       mov    eax, ERR_PERM_DENIED ; permission denied!
0000897B F9                   <1>       stc
0000897C C3                   <1>       retn
                              <1>
                              <1> loc_rmdir_get_last_cluster_2:
                              <1>       ; 29/12/2017
0000897D A3[64630100]         <1>       mov    [DelFile_FCluster], eax
                              <1>
                              <1>       ;mov   dx, [DirBuff_EntryCounter]
00008982 668B15[1C630100]     <1>       mov    dx, [FindFile_DirEntryNumber] ; 27/02/2016
00008989 668915[68630100]     <1>       mov    [DelFile_EntryCounter], dx
                              <1>
00008990 8B15[2D610100]       <1>       mov    edx, [DirBuff_Cluster]
00008996 8915[94630100]       <1>       mov    [RmDir_ParentDirCluster], edx
                              <1>
0000899C 893D[90630100]       <1>       mov    [RmDir_DirEntryOffset], edi
                              <1>
                              <1>       ; 01/03/2016
                              <1>       ;mov   dword [FAT_ClusterCounter], 0 ; Reset
                              <1>
                              <1> loc_rmdir_get_last_cluster_3:
000089A2 E89C390000           <1>       call   get_last_cluster
                              <1>         ;jc  loc_rmdir_cmd_failed
000089A7 721E                 <1>       jc     short loc_delete_sub_dir_retn ; 29/12/2017
                              <1>
000089A9 3B05[64630100]       <1>       cmp    eax, [DelFile_FCluster]
000089AF 7517                 <1>       jne    short loc_rmdir_multi_dir_clusters
                              <1>
000089B1 C605[8F630100]00     <1>       mov    byte [RmDir_MultiClusters], 0
```

```
000089B8 EB15                <1>       jmp    short pass_rmdir_multi_dir_clusters
                             <1> 
                             <1> loc_rmdir_check_fs_directory:
                             <1>       ; 29/12/2017
000089BA 807E04A1            <1>       cmp    byte [esi+LD_FSType], 0A1h
000089BE 75B6                <1>       jne    short loc_rmdir_permission_denied
                             <1> 
                             <1> loc_rmdir_delete_fs_directory:
000089C0 E876130000          <1>       call   delete_fs_directory
                             <1>       ;jnc   loc_print_deleted_message
000089C5 7300                <1>       jnc    short loc_delete_sub_dir_retn ; 29/12/2017
                             <1> 
                             <1>       ; EAX=0 -> Directory not empty !
                             <1>       ; EAX>0 -> Disk r/w error or another (misc) error
                             <1> 
                             <1>       ;or    eax, eax
                             <1>       ;jz    loc_rmdir_directory_not_empty_2
                             <1>       ;;stc
                             <1>       ;;jmp  loc_file_rw_cmd_failed
                             <1> 
                             <1> loc_delete_sub_dir_retn:
000089C7 C3                  <1>       retn
                             <1> 
                             <1> loc_rmdir_multi_dir_clusters:
000089C8 C605[8F630100]01    <1>       mov    byte [RmDir_MultiClusters], 1
                             <1> 
                             <1> pass_rmdir_multi_dir_clusters:
000089CF A3[98630100]        <1>       mov    [RmDir_DirLastCluster], eax
000089D4 890D[9C630100]      <1>       mov    [RmDir_PreviousCluster], ecx
                             <1> 
                             <1> loc_rmdir_load_fat_sub_directory:
000089DA E84F330000          <1>       call   load_FAT_sub_directory
                             <1>       ;jc    loc_rmdir_cmd_failed
000089DF 72E6                <1>       jc     short loc_delete_sub_dir_retn
                             <1> 
                             <1> loc_rmdir_find_last_dir_entry:
000089E1 56                  <1>       push   esi
000089E2 BE[86620100]        <1>       mov    esi, Dir_File_Name
000089E7 C6062A              <1>       mov    byte [esi], '*'
000089EA C646082A            <1>       mov    byte [esi+8], '*'
000089EE 31DB                <1>       xor    ebx, ebx ; Entry offset  = 0
                             <1> loc_rmdir_find_last_dir_entry_next:
000089F0 66B80008            <1>       mov    ax, 0800h ; Except volume/long names
000089F4 6631C9              <1>       xor    cx, cx ; 0 = Find a valid file or dir name
000089F7 E87A170000          <1>       call   find_directory_entry
000089FC 7225                <1>       jc     short loc_rmdir_empty_dir_cluster
000089FE 83FB01              <1>       cmp    ebx, 1
00008A01 771B                <1>       ja     short loc_rmdir_directory_not_empty_1
                             <1> loc_rmdir_dot_entry_check:
00008A03 80FD2E              <1>       cmp    ch, '.' ; The first char of the dir entry
00008A06 7516                <1>       jne    short loc_rmdir_directory_not_empty_1
00008A08 08DB                <1>       or     bl, bl
00008A0A 7506                <1>       jnz    short loc_rmdir_dotdot_entry_check
00008A0C 807F0120            <1>       cmp    byte [edi+1], 20h
00008A10 EB06                <1>       jmp    short pass_rmdir_dot_entry_check
                             <1> 
                             <1> loc_rmdir_dotdot_entry_check:
00008A12 66817F012E20        <1>       cmp    word [edi+1], '. '
                             <1> pass_rmdir_dot_entry_check:
00008A18 7504                <1>       jne    short loc_rmdir_directory_not_empty_1
00008A1A FEC3                <1>       inc    bl
00008A1C EBD2                <1>       jmp    short loc_rmdir_find_last_dir_entry_next
                             <1> 
                             <1> loc_rmdir_directory_not_empty_1:
00008A1E 58                  <1>       pop    eax ; pushed esi
00008A1F 31C0                <1>       xor    eax, eax ; 0
                             <1> loc_rmdir_directory_not_empty_2:
                             <1> loc_delete_sub_dir_stc_retn:
00008A21 F9                  <1>       stc
00008A22 C3                  <1>       retn
                             <1> 
                             <1> loc_rmdir_empty_dir_cluster:
00008A23 5E                  <1>       pop    esi
                             <1> 
                             <1> loc_rmdir_set_prev_cluster_dir_last_cluster:
00008A24 803D[8F630100]00    <1>       cmp    byte [RmDir_MultiClusters], 0
00008A2B 7613                <1>       jna    short loc_rmdir_unlink_dir_last_cluster
                             <1> 
00008A2D A1[9C630100]        <1>       mov    eax, [RmDir_PreviousCluster]
                             <1>       ;xor   ecx, ecx
00008A32 49                  <1>       dec    ecx ; FFFFFFFFh
00008A33 E83A340000          <1>       call   update_cluster
00008A38 7306                <1>       jnc    short loc_rmdir_unlink_dir_last_cluster
                             <1> 
                             <1>       ; 01/03/2016
                             <1>       ;cmp   eax, 1  ; eax = 0 -> end of cluster chain
                             <1>       ;cmc
                             <1>       ;jc    short loc_rmdir_cmd_failed
                             <1>       ;jmp   short loc_rmdir_save_fat_buffer
                             <1>       ; 29/12/2017
00008A3A 21C0                <1>       and    eax, eax
00008A3C 75E3                <1>       jnz    short loc_delete_sub_dir_stc_retn
00008A3E EB12                <1>       jmp    short loc_rmdir_save_fat_buffer
                             <1> 
                             <1> loc_rmdir_unlink_dir_last_cluster:
00008A40 A1[98630100]        <1>       mov    eax, [RmDir_DirLastCluster]
00008A45 31C9                <1>       xor    ecx, ecx ; 0
00008A47 E826340000          <1>       call   update_cluster
00008A4C 7327                <1>       jnc    short loc_rmdir_unlink_stc_retn_0Bh
                             <1>       ; Because of it is the last cluster
                             <1>       ; 'update_cluster' must return with eocc error
00008A4E 09C0                <1>       or     eax, eax
                             <1>       ;jz    short loc_rmdir_save_fat_buffer ; eocc
                             <1>       ;stc
                             <1>         ;jmp     short loc_rmdir_cmd_failed
                             <1>       ; 29/12/2017
00008A50 75CF                <1>       jnz    short loc_delete_sub_dir_stc_retn
                             <1>
                             <1> loc_rmdir_save_fat_buffer:
00008A52 803D[16610100]02    <1>       cmp    byte [FAT_BuffValidData], 2
00008A59 7528                <1>       jne    short loc_rmdir_calculate_FAT_freespace
00008A5B E8CF360000          <1>       call   save_fat_buffer
                             <1>       ;jc    short loc_rmdir_cmd_failed
                             <1>       ; 29/12/2017
00008A60 7219                <1>       jc     short loc_rmdir_unlink_error_retn
                             <1>
                             <1>       ; 01/03/2016
00008A62 803D[8F630100]00    <1>       cmp    byte [RmDir_MultiClusters], 0
00008A69 7618                <1>       jna    short loc_rmdir_calculate_FAT_freespace
                             <1>
00008A6B A1[64630100]        <1>       mov    eax, [DelFile_FCluster]
00008A70 E92DFFFFFF          <1>         jmp     loc_rmdir_get_last_cluster_3
                             <1>
                             <1> loc_rmdir_unlink_stc_retn_0Bh:
                             <1>       ; 15/10/2016 (0Bh -> 28)
00008A75 B81C000000          <1>       mov    eax, ERR_INV_FORMAT ; 28 = Invalid format
                             <1> loc_rmdir_unlink_stc_retn:
00008A7A F9                  <1>       stc
                             <1> loc_rmdir_unlink_error_retn:
00008A7B C3                  <1>       retn
                             <1>
                             <1> loc_rmdir_delete_short_name_invalid_data:
00008A7C B81D000000          <1>       mov    eax, 29 ; Invalid data (15/10/2016)
                             <1>       ;stc
                             <1>         ;jmp loc_rmdir_cmd_failed
                             <1>       ; 29/12/2017
00008A81 EBF7                <1>       jmp    short loc_rmdir_unlink_stc_retn
                             <1>
                             <1> loc_rmdir_calculate_FAT_freespace:
                             <1>       ;mov   eax, [FAT_ClusterCounter]
                             <1>       ; 29/12/2017
00008A83 29C0                <1>       sub    eax, eax ; 0
00008A85 8705[1E610100]      <1>       xchg   eax, [FAT_ClusterCounter]
                             <1>       ;
00008A8B 66BB01FF            <1>       mov    bx, 0FF01h
                             <1>       ; BL = 1 -> Add EAX to free space count
                             <1>       ; BH = FFh ->
                             <1>       ; ESI = Logical DOS Drive Description Table address
00008A8F E830370000          <1>       call   calculate_fat_freespace
                             <1>
00008A94 21C9                <1>       and    ecx, ecx ; ecx = 0 -> valid free sector count
00008A96 7409                <1>       jz     short loc_rmdir_delete_short_name_continue
                             <1>
                             <1> loc_rmdir_recalculate_FAT_freespace:
00008A98 66BB00FF            <1>         mov     bx, 0FF00h ; BL = 0 -> Recalculate free space
00008A9C E823370000          <1>       call   calculate_fat_freespace
                             <1>
                             <1> loc_rmdir_delete_short_name_continue:
00008AA1 A1[94630100]        <1>       mov    eax, [RmDir_ParentDirCluster]
00008AA6 83F802              <1>       cmp    eax, 2
00008AA9 7309                <1>       jnb    short loc_rmdir_del_short_name_load_sub_dir
00008AAB E8F3310000          <1>       call   load_FAT_root_directory
                             <1>       ;jc    loc_file_rw_cmd_failed
                             <1>       ; 29/12/2017
00008AB0 72C9                <1>       jc     short loc_rmdir_unlink_error_retn
00008AB2 EB07                <1>       jmp    short loc_rmdir_del_short_name_ld_chk_fclust
                             <1>
                             <1> loc_rmdir_del_short_name_load_sub_dir:
00008AB4 E875320000          <1>       call   load_FAT_sub_directory
                             <1>       ;jc    loc_file_rw_cmd_failed
                             <1>       ; 29/12/2017
00008AB9 72C0                <1>       jc     short loc_rmdir_unlink_error_retn
                             <1>
                             <1> loc_rmdir_del_short_name_ld_chk_fclust:
00008ABB 0FB73D[90630100]    <1>       movzx  edi, word [RmDir_DirEntryOffset]
00008AC2 81C700000800        <1>       add    edi, Directory_Buffer
                             <1>
00008AC8 668B4714            <1>       mov    ax, [edi+20] ; First Cluster High Word
00008ACC C1E010              <1>       shl    eax, 16
00008ACF 668B471A            <1>       mov    ax, [edi+26] ; First Cluster Low Word
                             <1>         ; Not necessary...
00008AD3 3B05[64630100]      <1>       cmp    eax, [DelFile_FCluster]
00008AD9 75A1                <1>       jne    short loc_rmdir_delete_short_name_invalid_data
                             <1>       ;
00008ADB C607E5              <1>       mov    byte [edi], 0E5h ; 'Deleted' sign
                             <1>       ; 27/02/2016
                             <1>       ; TRDOS v1 has a bug here! it does not set
                             <1>       ; 'DirBuff_ValidData' to 2; as result of this bug,
                             <1>       ; 'save_directory_buffer' would not save the change !
00008ADE C605[28610100]02    <1>       mov    byte [DirBuff_ValidData], 2 ; change sign
                             <1>       ;
00008AE5 E8AE1D0000          <1>       call   save_directory_buffer
                             <1>       ;jc    loc_file_rw_cmd_failed
                             <1>       ; 29/12/2017
00008AEA 728F                <1>       jc     short loc_rmdir_unlink_error_retn
                             <1>
                             <1> loc_rmdir_del_long_name:
00008AEC 0FB615[6A630100]    <1>       movzx  edx, byte [DelFile_LNEL]
00008AF3 08D2                <1>       or     dl, dl
00008AF5 7410                <1>       jz     short loc_rmdir_update_parent_dir_lmdt
                             <1>
00008AF7 0FB705[68630100]    <1>       movzx  eax, word [DelFile_EntryCounter]
00008AFE 29D0                <1>       sub    eax, edx
                             <1>       ; 29/12/2017
00008B00 7205                <1>       jc     short loc_rmdir_update_parent_dir_lmdt
                             <1>
```

```
                                  <1>       ; EAX = Directory Entry Number of the long name last entry
00008B02 E8EF1E0000               <1>       call   delete_longname
                                  <1>
                                  <1> loc_rmdir_update_parent_dir_lmdt:
00008B07 E8271E0000               <1>       call   update_parent_dir_lmdt
                                  <1>       ;jc    short loc_file_rw_cmd_failed
                                  <1>       ; 29/12/2017
                                  <1>       ;jc    short loc_rmdir_unlink_error_retn
                                  <1>
                                  <1> loc_delete_sub_directory_ok:
                                  <1>       ; 29/12/2017
00008B0C 31C0                     <1>       xor    eax, eax ;  0 ;  cf = 0
00008B0E C3                       <1>       retn
                                  <1>
                                  <1>
                                  <1> delete_file:
                                  <1>       ; 29/02/2016
                                  <1>       ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)
                                  <1>       ; 09/08/2010 (CMD_INTR.ASM, 'cmp_cmd_del')
                                  <1>       ; 28/02/2010
                                  <1>
                                  <1> get_delfile_fchar:
                                  <1>       ; esi = file name
00008B0F 803E20                   <1>       cmp    byte [esi], 20h
00008B12 7701                     <1>         ja   short loc_delfile_parse_path_name
                                  <1>
                                  <1> loc_delfile_nofilename_retn:
00008B14 C3                       <1>       retn
                                  <1>
                                  <1> loc_delfile_parse_path_name:
00008B15 BF[A2620100]             <1>       mov    edi, FindFile_Drv
00008B1A E815190000               <1>       call   parse_path_name
00008B1F 0F822EF2FFFF             <1>       jc     loc_cmd_failed
                                  <1>
                                  <1> loc_delfile_check_filename_exists:
00008B25 BE[E4620100]             <1>       mov    esi, FindFile_Name
00008B2A 803E20                   <1>       cmp    byte [esi], 20h
00008B2D 0F8620F2FFFF             <1>       jna    loc_cmd_failed
00008B33 8935[60630100]           <1>       mov    [DelFile_FNPointer], esi
                                  <1>
                                  <1> loc_delfile_drv:
00008B39 8A15[A2620100]           <1>       mov    dl, [FindFile_Drv]
00008B3F 8A35[FE580100]           <1>       mov    dh, [Current_Drv]
00008B45 8835[5E610100]           <1>       mov    [RUN_CDRV], dh
00008B4B 38F2                     <1>       cmp    dl, dh
00008B4D 740B                     <1>       je     short loc_delfile_change_directory
                                  <1>
00008B4F E86CE3FFFF               <1>       call   change_current_drive
00008B54 0F8237FBFFFF             <1>       jc     loc_file_rw_cmd_failed
                                  <1>
                                  <1> loc_delfile_change_directory:
00008B5A 803D[A3620100]20         <1>       cmp    byte [FindFile_Directory], 20h
00008B61 7618                     <1>       jna    short loc_delfile_find
                                  <1>
00008B63 FE05[D30C0100]           <1>       inc    byte [Restore_CDIR]
00008B69 BE[A3620100]             <1>       mov    esi, FindFile_Directory
00008B6E 30E4                     <1>       xor    ah, ah ; CD_COMMAND sign -> 0
00008B70 E8A9120000               <1>       call   change_current_directory
00008B75 0F8216FBFFFF             <1>       jc     loc_file_rw_cmd_failed
                                  <1>
                                  <1> ;loc_delfile_change_prompt_dir_string:
                                  <1>       ;call  change_prompt_dir_string
                                  <1>
                                  <1> loc_delfile_find:
                                  <1>       ;mov   esi, FindFile_Name
00008B7B 8B35[60630100]           <1>       mov    esi, [DelFile_FNPointer]
00008B81 66B80018                 <1>       mov    ax, 1800h ; Except volume label and dirs
00008B85 E8D9F6FFFF               <1>       call   find_first_file
00008B8A 0F8201FBFFFF             <1>       jc     loc_file_rw_cmd_failed
                                  <1>
                                  <1> loc_delfile_ambgfn_check:
00008B90 6621D2                   <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
00008B93 740B                     <1>       jz     short loc_delfile_found
                                  <1>
                                  <1> loc_file_not_found:
00008B95 B802000000               <1>       mov    eax, 2 ; File not found sign
00008B9A F9                       <1>       stc
00008B9B E9F1FAFFFF               <1>       jmp    loc_file_rw_cmd_failed
                                  <1>
                                  <1> loc_delfile_found:
00008BA0 80E307                   <1>       and    bl, 07h ; Attributes
00008BA3 0F85F2FAFFFF             <1>         jnz     loc_permission_denied
                                  <1>
                                  <1> ;loc_delfile_found_save_lnel:
                                  <1> ;     mov    [DelFile_LNEL], bh ; Long name entry length (if > 0)
                                  <1>
                                  <1> loc_delfile_ask_for_delete:
00008BA9 57                       <1>       push   edi ; * (29/02/2016)
                                  <1>
00008BAA BE[A3110100]             <1>       mov    esi, Msg_DoYouWantDelete
00008BAF E8A9D7FFFF               <1>       call   print_msg
00008BB4 8B35[60630100]           <1>       mov    esi, [DelFile_FNPointer]
00008BBA E89ED7FFFF               <1>       call   print_msg
00008BBF BE[57110100]             <1>       mov    esi, Msg_YesNo
00008BC4 E894D7FFFF               <1>       call   print_msg
                                  <1>
                                  <1> loc_delfile_ask_again:
00008BC9 30E4                     <1>       xor    ah, ah
00008BCB E84680FFFF               <1>       call   int16h
00008BD0 3C1B                     <1>       cmp    al, 1Bh
                                  <1>       ;je    short loc_do_not_delete_file
00008BD2 7449                     <1>       je     short loc_delfile_y_n_escape ; 06/03/2016
00008BD4 24DF                     <1>       and    al, 0DFh
```

```
00008BD6 A2[61110100]        <1>       mov    [Y_N_nextline], al
00008BDB 3C59                <1>       cmp    al, 'Y'
00008BDD 7404                <1>       je     short loc_yes_delete_file
00008BDF 3C4E                <1>       cmp    al, 'N'
00008BE1 75E6                <1>       jne    short loc_delfile_ask_again
                             <1>
                             <1> loc_do_not_delete_file:
                             <1> loc_yes_delete_file:
00008BE3 E8E2FBFFFF          <1>       call   y_n_answer ; 29/12/2017
00008BE8 5F                  <1>       pop    edi ; * (29/02/2016)
                             <1>       ;cmp   al, 'Y' ; 'yes'
                             <1>       ;cmc
                             <1>         ;jnc loc_file_rw_restore_retn
00008BE9 3C4E                <1>       cmp    al, 'N' ; 'no'
00008BEB 0F84A0FAFFFF        <1>         je   loc_file_rw_restore_retn
                             <1>
                             <1> loc_delete_file:
00008BF1 8A3D[A2620100]      <1>       mov    bh, [FindFile_Drv]
                             <1>       ;mov   bl, [DelFile_LNEL]
00008BF7 8A1D[F1620100]      <1>       mov    bl, [FindFile_LongNameEntryLength]
                             <1>       ;mov   cx, [DirBuff_EntryCounter]
00008BFD 668B0D[1C630100]    <1>       mov    cx, [FindFile_DirEntryNumber]
                             <1>       ; (*) EDI = Directory buffer entry offset/address
00008C04 E8D71F0000          <1>       call   remove_file ; (FILE.ASM, 'proc_delete_file')
00008C09 0F8378FAFFFF        <1>       jnc    loc_print_deleted_message
                             <1>
                             <1>       ;cmp   al, 05h
00008C0F 3C0B                <1>       cmp    al, ERR_PERM_DENIED  ; 29/12/2017 (5 -> 11)
00008C11 0F8484FAFFFF        <1>       je     loc_permission_denied
00008C17 F9                  <1>       stc
00008C18 E974FAFFFF          <1>       jmp    loc_file_rw_cmd_failed
                             <1>
                             <1> loc_delfile_y_n_escape:
00008C1D B04E                <1>       mov    al, 'N' ; 'no'
00008C1F EBC2                <1>       jmp    short loc_do_not_delete_file
                             <1>
                             <1> set_file_attributes:
                             <1>       ; 06/03/2016
                             <1>       ; 04/03/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 10/07/2010 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_attrib')
                             <1>       ; 23/05/2010
                             <1>       ; 17/12/2000 (P2000.ASM)
                             <1>
                             <1>       ; esi = file or directory name
00008C21 6631C0              <1>       xor    ax, ax
00008C24 66A3[F4110100]      <1>       mov    [Attr_Chars], ax
00008C2A A2[B8630100]        <1>       mov    [Attributes], al
                             <1>
                             <1> get_attrib_fchar:
                             <1>       ; esi = file name
00008C2F 8A06                <1>       mov    al, [esi]
00008C31 3C20                <1>       cmp    al, 20h
00008C33 7623                <1>       jna    short loc_attr_file_nofilename_retn
                             <1>
                             <1> loc_scan_attrib_params:
00008C35 3C2D                <1>       cmp    al, '-'
00008C37 0F871C010000        <1>       ja     loc_attr_file_parse_path_name
00008C3D 7408                <1>       je     short loc_attr_space
                             <1>
00008C3F 3C2B                <1>       cmp    al, '+'
00008C41 0F850CF1FFFF        <1>       jne    loc_cmd_failed
                             <1>
                             <1> loc_attr_space:
00008C47 8A6601              <1>       mov    ah, [esi+1]
00008C4A 80FC20              <1>       cmp    ah, 20h
00008C4D 770A                <1>       ja     short pass_attr_space
00008C4F 0F82FEF0FFFF        <1>       jb     loc_cmd_failed
00008C55 46                  <1>       inc    esi
00008C56 EBEF                <1>       jmp    short loc_attr_space
                             <1>
                             <1> loc_attr_file_nofilename_retn:
00008C58 C3                  <1>       retn
                             <1>
                             <1> pass_attr_space:
00008C59 80E4DF              <1>       and    ah, 0DFh
00008C5C 80FC53              <1>       cmp    ah, 'S'
00008C5F 0F87EEF0FFFF        <1>       ja     loc_cmd_failed
00008C65 7204                <1>       jb     short pass_attr_system
00008C67 B404                <1>       mov    ah, 04h   ; System
00008C69 EB21                <1>       jmp    short pass_attr_archive
                             <1>
                             <1> pass_attr_system:
00008C6B 80FC48              <1>       cmp    ah, 'H'
00008C6E 7706                <1>       ja     short pass_attr_hidden
00008C70 7213                <1>       jb     short pass_attr_read_only
00008C72 B402                <1>       mov    ah, 02h    ; Hidden
00008C74 EB16                <1>       jmp    short pass_attr_archive
                             <1>
                             <1> pass_attr_hidden:
00008C76 80FC52              <1>       cmp    ah, 'R'
00008C79 0F87D4F0FFFF        <1>       ja     loc_cmd_failed
00008C7F 7204                <1>       jb     short pass_attr_read_only ; Read only
00008C81 B401                <1>       mov    ah, 01h
00008C83 EB07                <1>       jmp    short pass_attr_archive
                             <1>
                             <1> pass_attr_read_only:
00008C85 80FC41              <1>       cmp    ah, 'A'
00008C88 753B                <1>       jne    short loc_chk_attr_enter
00008C8A B420                <1>       mov    ah, 20h    ; Archive
                             <1>
                             <1> pass_attr_archive:
00008C8C 3C2D                <1>       cmp    al, '-'
00008C8E 7508                <1>       jne    short pass_reducing_attributes
```

```
00008C90 0825[F4110100]      <1>       or     [Attr_Chars], ah
00008C96 EB06                <1>       jmp    short loc_change_attributes_inc
                             <1> 
                             <1> pass_reducing_attributes:
00008C98 0825[F5110100]      <1>       or     [Attr_Chars+1], ah
                             <1> 
                             <1> loc_change_attributes_inc:
00008C9E 46                  <1>       inc    esi
00008C9F 8A6601              <1>       mov    ah, [esi+1]
00008CA2 80FC20              <1>       cmp    ah, 20h
00008CA5 7227                <1>       jb     short pass_change_attr
00008CA7 74F5                <1>       je     short loc_change_attributes_inc
00008CA9 80FC2D              <1>       cmp    ah, '-'
00008CAC 770D                <1>       ja     short loc_chk_next_attr_char1
00008CAE 7405                <1>       je     short loc_chk_next_attr_char0
00008CB0 80FC2B              <1>       cmp    ah, '+'
00008CB3 7506                <1>       jne    short loc_chk_next_attr_char1
                             <1> 
                             <1> loc_chk_next_attr_char0:
00008CB5 46                  <1>       inc    esi
00008CB6 668B06              <1>       mov    ax, [esi]
00008CB9 EB9E                <1>       jmp    short pass_attr_space
                             <1> 
                             <1> loc_chk_next_attr_char1:
00008CBB 803E2D              <1>       cmp    byte [esi], '-'
00008CBE 7799                <1>       ja     short pass_attr_space
00008CC0 E988000000          <1>         jmp      loc_attr_file_check_fname_fchar
                             <1> 
                             <1> loc_chk_attr_enter:
00008CC5 80FC0D              <1>       cmp    ah, 0Dh
00008CC8 0F8585F0FFFF        <1>       jne    loc_cmd_failed
                             <1> 
                             <1> pass_change_attr:
00008CCE A0[F4110100]        <1>       mov    al, [Attr_Chars]
00008CD3 F6D0                <1>       not    al
00008CD5 2005[B8630100]      <1>       and    [Attributes], al
00008CDB A0[F5110100]        <1>       mov    al, [Attr_Chars+1]
00008CE0 0805[B8630100]      <1>       or     [Attributes], al
                             <1> 
                             <1> loc_show_attributes:
00008CE6 BE[6F190100]        <1>       mov    esi, nextline
00008CEB E86DD6FFFF          <1>       call   print_msg
                             <1> 
                             <1> loc_show_attributes_no_nextline:
00008CF0 C705[F4110100]4E4F- <1>       mov    dword [Attr_Chars], 'NORM'
00008CF8 524D                <1>
00008CFA 66C705[F8110100]41- <1>       mov    word [Attr_Chars+4], 'AL'
00008D02 4C                  <1>
00008D03 BE[F4110100]        <1>       mov    esi, Attr_Chars
00008D08 A0[B8630100]        <1>       mov    al, [Attributes]
00008D0D A804                <1>       test   al, 04h
00008D0F 7406                <1>       jz     short pass_put_attr_s
00008D11 66C7065300          <1>       mov    word [esi], 0053h     ; S
00008D16 46                  <1>       inc    esi
                             <1> 
                             <1> pass_put_attr_s:
00008D17 A802                <1>       test   al, 02h
00008D19 7406                <1>       jz     short pass_put_attr_h
00008D1B 66C7064800          <1>       mov    word [esi], 0048h     ; H
00008D20 46                  <1>       inc    esi
                             <1> 
                             <1> pass_put_attr_h:
00008D21 A801                <1>       test   al, 01h
00008D23 7406                <1>       jz     short pass_put_attr_r
00008D25 66C7065200          <1>       mov    word [esi], 0052h     ; R
00008D2A 46                  <1>       inc    esi
                             <1> 
                             <1> pass_put_attr_r:
00008D2B 3C20                <1>       cmp    al, 20h
00008D2D 7205                <1>       jb     short pass_put_attr_a
00008D2F 66C7064100          <1>       mov    word [esi], 0041h     ; A
                             <1> 
                             <1> pass_put_attr_a:
00008D34 BE[E7110100]        <1>       mov    esi, Str_Attributes
00008D39 E81FD6FFFF          <1>       call   print_msg
00008D3E BE[6F190100]        <1>       mov    esi, nextline
00008D43 E815D6FFFF          <1>       call   print_msg
00008D48 E944F9FFFF          <1>       jmp    loc_file_rw_restore_retn
                             <1> 
                             <1> loc_attr_file_check_fname_fchar:
00008D4D 46                  <1>       inc    esi
00008D4E 803E20              <1>       cmp    byte [esi], 20h
00008D51 74FA                <1>       je     short loc_attr_file_check_fname_fchar
00008D53 0F8275FFFFFF        <1>         jb      pass_change_attr
                             <1> 
                             <1> loc_attr_file_parse_path_name:
00008D59 BF[A2620100]        <1>       mov    edi, FindFile_Drv
00008D5E E8D1160000          <1>       call   parse_path_name
00008D63 0F82EAEFFFFF        <1>       jc     loc_cmd_failed
                             <1> 
                             <1> loc_attr_file_check_filename_exists:
00008D69 BE[E4620100]        <1>       mov    esi, FindFile_Name
00008D6E 803E20              <1>       cmp    byte [esi], 20h
00008D71 0F86DCEFFFFF        <1>       jna    loc_cmd_failed
00008D77 8935[60630100]      <1>       mov    [DelFile_FNPointer], esi
                             <1> 
                             <1> loc_attr_file_drv:
00008D7D 8A35[FE580100]      <1>       mov    dh, [Current_Drv]
00008D83 8835[5E610100]      <1>       mov    [RUN_CDRV], dh
                             <1> 
00008D89 8A15[A2620100]      <1>       mov    dl, [FindFile_Drv]
00008D8F 38F2                <1>       cmp    dl, dh
00008D91 740B                <1>       je     short loc_attr_file_change_directory
```

```
                             <1> 
00008D93 E828E1FFFF          <1>       call   change_current_drive
00008D98 0F82F3F8FFFF        <1>       jc     loc_file_rw_cmd_failed
                             <1> 
                             <1> loc_attr_file_change_directory:
00008D9E 803D[A3620100]20    <1>         cmp     byte [FindFile_Directory], 20h
00008DA5 7618                <1>       jna    short loc_attr_file_find
                             <1> 
00008DA7 FE05[D30C0100]      <1>       inc    byte [Restore_CDIR]
                             <1> 
00008DAD BE[A3620100]        <1>       mov    esi, FindFile_Directory
00008DB2 30E4                <1>       xor    ah, ah ; CD_COMMAND sign -> 0
00008DB4 E865100000          <1>       call   change_current_directory
00008DB9 0F82D2F8FFFF        <1>       jc     loc_file_rw_cmd_failed
                             <1> 
                             <1> ;loc_attr_file_change_prompt_dir_string:
                             <1>       ;call  change_prompt_dir_string
                             <1> 
                             <1> loc_attr_file_find:
                             <1>       ;mov   esi, FindFile_Name
00008DBF 8B35[60630100]      <1>       mov    esi, [DelFile_FNPointer]
00008DC5 66B80008            <1>       mov    ax, 0800h ; Except volume labels
00008DC9 E895F4FFFF          <1>       call   find_first_file
00008DCE 0F82BDF8FFFF        <1>       jc     loc_file_rw_cmd_failed
                             <1> 
                             <1> loc_attr_file_ambgfn_check:
00008DD4 6609D2              <1>       or     dx, dx ; Ambiguous filename chars used sign (DX>0)
                             <1>       ;      (Note: It was BX in TRDOS v1)
                             <1>       ;jz    short loc_attr_file_found
00008DD7 0F85B8FDFFFF        <1>         jnz     loc_file_not_found ; 06/03/2016
                             <1> 
                             <1>       ;mov   eax, 2 ; File not found sign
                             <1>       ;stc
                             <1>       ;jmp   loc_file_rw_cmd_failed
                             <1> 
                             <1> loc_attr_file_found:
                             <1>       ; EDI = Directory buffer entry offset/address
                             <1>       ; BL = File (or Directory) Attributes
                             <1>       ;      (Note: It was 'CL' in TRDOS v1)
                             <1>       ; mov  bl, [EDI+0Bh]
                             <1> 
00008DDD 66833D[F4110100]00  <1>       cmp    word [Attr_Chars], 0
00008DE5 770B                <1>       ja     short loc_attr_file_change_attributes
00008DE7 881D[B8630100]      <1>       mov    [Attributes], bl
00008DED E9F4FEFFFF          <1>       jmp    loc_show_attributes
                             <1> 
                             <1> loc_attr_file_change_attributes:
00008DF2 A0[F4110100]        <1>       mov    al, [Attr_Chars]
00008DF7 F6D0                <1>       not    al
00008DF9 20C3                <1>       and    bl, al
00008DFB A0[F5110100]        <1>       mov    al, [Attr_Chars+1]
00008E00 08C3                <1>       or     bl, al
                             <1> 
00008E02 66817F0CA101        <1>       cmp    word [edi+DirEntry_NTRes], 01A1h ; Singlix FS
00008E08 741D                <1>       je     short loc_attr_file_fs_check
                             <1> 
00008E0A 881D[B8630100]      <1>       mov    [Attributes], bl
00008E10 885F0B              <1>       mov    [edi+0Bh], bl    ; Attributes (New!)
                             <1> 
                             <1>       ; 04/03/2016
                             <1>       ; TRDOS v1 has a bug here! it does not set
                             <1>       ; 'DirBuff_ValidData' to 2; as result of this bug,
                             <1>       ; 'save_directory_buffer' would not save the new attributes !
                             <1> 
00008E13 C605[28610100]02    <1>       mov    byte [DirBuff_ValidData], 2
                             <1> 
00008E1A E8791A0000          <1>       call   save_directory_buffer
00008E1F 0F826CF8FFFF        <1>       jc     loc_file_rw_cmd_failed
                             <1> 
00008E25 EB33                <1>       jmp    short loc_print_attr_changed_message
                             <1> 
                             <1> loc_attr_file_fs_check:
00008E27 29C0                <1>       sub    eax, eax
00008E29 8A25[26610100]      <1>         mov     ah, [DirBuff_DRV]
00008E2F BE00010900          <1>       mov    esi, Logical_DOSDisks
00008E34 01C6                <1>         add     esi, eax
00008E36 807E04A1            <1>         cmp     byte [esi+LD_FSType], 0A1h
00008E3A 7309                <1>       jnc    short loc_attr_file_change_fs_file_attributes
                             <1>       ; 29/12/2017 (0Dh -> 29)
00008E3C 66B81D00            <1>       mov    ax, 29 ; Invalid Data
00008E40 E94CF8FFFF          <1>       jmp    loc_file_rw_cmd_failed
                             <1> 
                             <1> loc_attr_file_change_fs_file_attributes:
                             <1>       ; BL = New MS-DOS File Attributes
00008E45 88D8                <1>       mov    al, bl ; File/Directory Attributes
00008E47 30E4                <1>       xor    ah, ah ; Attributes in MS-DOS format sign
00008E49 E873050000          <1>       call   change_fs_file_attributes
00008E4E 0F823DF8FFFF        <1>       jc     loc_file_rw_cmd_failed
                             <1> 
00008E54 881D[B8630100]      <1>       mov    [Attributes], bl
                             <1> 
                             <1> loc_print_attr_changed_message:
00008E5A BE[E2110100]        <1>       mov    esi, Msg_New
00008E5F E8F9D4FFFF          <1>       call   print_msg
00008E64 E987FEFFFF          <1>       jmp    loc_show_attributes_no_nextline
                             <1> 
                             <1> rename_file:
                             <1>       ; 13/11/2017
                             <1>       ; 06/11/2016
                             <1>       ; 05/11/2016
                             <1>       ; 16/10/2016
                             <1>       ; 08/03/2016
                             <1>       ; 06/03/2016 (TRDOS 386 = TRDOS v2.0)
```

```
                                 <1>       ; 20/11/2010 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_rename')
                                 <1>       ; 16/11/2010
                                 <1>
                                 <1> get_rename_source_fchar:
                                 <1>       ; esi = file name
00008E69 803E20                  <1>       cmp    byte [esi], 20h
00008E6C 7614                    <1>         jna  short loc_rename_nofilename_retn
                                 <1>
00008E6E 8935[E0630100]          <1>       mov    [SourceFilePath], esi
                                 <1>
                                 <1> rename_scan_source_file:
00008E74 46                      <1>       inc    esi
00008E75 803E20                  <1>       cmp    byte [esi], 20h
00008E78 7409                    <1>       je     short rename_scan_destination_file_1
                                 <1>       ;jb    short loc_rename_nofilename_retn
00008E7A 0F82D3EEFFFF            <1>       jb     loc_cmd_failed
00008E80 EBF2                    <1>       jmp    short rename_scan_source_file
                                 <1>
                                 <1> loc_rename_nofilename_retn: ; 08/03/2016
00008E82 C3                      <1>       retn
                                 <1>
                                 <1> rename_scan_destination_file_1:
00008E83 C60600                  <1>       mov    byte [esi], 0
                                 <1>
                                 <1> rename_scan_destination_file_2:
00008E86 46                      <1>       inc    esi
00008E87 803E20                  <1>       cmp    byte [esi], 20h
00008E8A 74FA                    <1>       je     short rename_scan_destination_file_2
                                 <1>       ;jb    short loc_rename_nofilename_retn
00008E8C 0F82C1EEFFFF            <1>       jb     loc_cmd_failed
                                 <1>
00008E92 8935[E4630100]          <1>       mov    [DestinationFilePath], esi
                                 <1>
                                 <1> rename_scan_destination_file_3:
00008E98 46                      <1>       inc    esi
00008E99 803E20                  <1>       cmp    byte [esi], 20h
00008E9C 77FA                    <1>       ja     short rename_scan_destination_file_3
                                 <1>
00008E9E C60600                  <1>       mov    byte [esi], 0
                                 <1>
                                 <1> loc_rename_save_current_drive:
00008EA1 8A35[FE580100]          <1>       mov    dh, [Current_Drv]
00008EA7 8835[5E610100]          <1>       mov    byte [RUN_CDRV], dh
                                 <1>
                                 <1> loc_rename_sf_parse_path_name:
00008EAD 8B35[E0630100]          <1>       mov    esi, [SourceFilePath]
00008EB3 BF[A2620100]            <1>       mov    edi, FindFile_Drv
00008EB8 E877150000              <1>       call   parse_path_name
00008EBD 0F8290EEFFFF            <1>       jc     loc_cmd_failed
                                 <1>
                                 <1> loc_rename_sf_check_filename_exists:
00008EC3 BE[E4620100]            <1>       mov    esi, FindFile_Name
00008EC8 803E20                  <1>       cmp    byte [esi], 20h
00008ECB 0F8682EEFFFF            <1>       jna    loc_cmd_failed
                                 <1>
                                 <1>       ;mov   [DelFile_FNPointer], esi
                                 <1>
                                 <1> loc_rename_sf_drv:
                                 <1>       ;mov   dh, [Current_Drv]
                                 <1>       ;mov   [RUN_CDRV], dh
                                 <1>
00008ED1 8A15[A2620100]          <1>       mov    dl, [FindFile_Drv]
00008ED7 38F2                    <1>       cmp    dl, dh ; dh = [Current_Drv]
00008ED9 740B                    <1>       je     short rename_sf_change_directory
                                 <1>
00008EDB E8E0DFFFFF              <1>       call   change_current_drive
00008EE0 0F82ABF7FFFF            <1>       jc     loc_file_rw_cmd_failed
                                 <1>
                                 <1> rename_sf_change_directory:
00008EE6 803D[A3620100]20        <1>       cmp    byte [FindFile_Directory], 20h
00008EED 7618                    <1>       jna    short rename_sf_find
                                 <1>
00008EEF FE05[D30C0100]          <1>       inc    byte [Restore_CDIR]
00008EF5 BE[A3620100]            <1>       mov    esi, FindFile_Directory
00008EFA 30E4                    <1>       xor    ah, ah ; CD_COMMAND sign -> 0
00008EFC E81D0F0000              <1>       call   change_current_directory
00008F01 0F828AF7FFFF            <1>       jc     loc_file_rw_cmd_failed
                                 <1>
                                 <1> ;rename_sf_change_prompt_dir_string:
                                 <1>       ;call  change_prompt_dir_string
                                 <1>
                                 <1> rename_sf_find:
                                 <1>       ;mov   esi, [DelFile_FNPointer]
00008F07 BE[E4620100]            <1>       mov    esi, FindFile_Name
                                 <1>
00008F0C 66B80008                <1>       mov    ax, 0800h ; Except volume labels
00008F10 E84EF3FFFF              <1>       call   find_first_file
00008F15 0F8276F7FFFF            <1>       jc     loc_file_rw_cmd_failed
                                 <1>
                                 <1> loc_rename_sf_ambgfn_check:
00008F1B 6621D2                  <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
                                 <1>       ;      (Note: It was BX in TRDOS v1)
                                 <1>       ;jz    short loc_rename_sf_found
00008F1E 0F8571FCFFFF            <1>       jnz    loc_file_not_found
                                 <1>
                                 <1>       ;mov   eax, 2 ; File not found sign
                                 <1>       ;stc
                                 <1>       ;jmp   loc_file_rw_cmd_failed
                                 <1>
                                 <1> loc_rename_sf_found:
                                 <1>       ; EDI = Directory buffer entry offset/address
                                 <1>       ; BL = File (or Directory) Attributes
                                 <1>       ;      (Note: It was 'CL' in TRDOS v1)
```

```
                             <1>       ; mov  bl, [EDI+0Bh]
                             <1>
00008F24 F6C307              <1>       test   bl, 07h ; Attributes, S-H-R
00008F27 0F856EF7FFFF        <1>       jnz    loc_permission_denied
                             <1>
00008F2D BE[A2620100]        <1>         mov     esi, FindFile_Drv
00008F32 BF[E8630100]        <1>         mov     edi, SourceFile_Drv
00008F37 B920000000          <1>       mov    ecx, 32
00008F3C F3A5                <1>       rep    movsd
                             <1>
                             <1> loc_rename_df_parse_path_name:
00008F3E 8B35[E4630100]      <1>       mov    esi, [DestinationFilePath]
00008F44 BF[A2620100]        <1>       mov    edi, FindFile_Drv
00008F49 E8E6140000          <1>       call   parse_path_name
00008F4E 7219                <1>       jc     short loc_rename_df_cmd_failed
                             <1>
                             <1>       ;mov   dh, [RUN_CDRV]
00008F50 8A35[FE580100]      <1>       mov    dh, [Current_Drv]
                             <1>
                             <1>       ; 'rename' command is valid only for same dos drive and same dir!
                             <1>       ; ('move' command must be used if source file and destination file
                             <1>       ; directories are not same!)
00008F56 8A15[A2620100]      <1>       mov    dl, [FindFile_Drv]
00008F5C 38F2                <1>       cmp    dl, dh ; are source and destination drives different ?!
00008F5E 7509                <1>       jne    short loc_rename_df_cmd_failed ; yes!
                             <1>
                             <1> rename_df_check_dirname_exists:
00008F60 803D[A3620100]00    <1>       cmp    byte [FindFile_Directory], 0
00008F67 760B                <1>       jna    short rename_df_check_filename_exists
                             <1>
                             <1>       ; different source file and destination file directories !
                             <1> loc_rename_df_cmd_failed:
00008F69 B801000000          <1>       mov    eax, 1 ; TRDOS 'Bad command or file name' error
00008F6E F9                  <1>       stc
00008F6F E91DF7FFFF          <1>       jmp    loc_file_rw_cmd_failed
                             <1>
                             <1> rename_df_check_filename_exists:
00008F74 BE[E4620100]        <1>       mov    esi, FindFile_Name
00008F79 E8A3F6FFFF          <1>       call   check_filename
00008F7E 0F82BFF7FFFF        <1>       jc     loc_mkdir_invalid_dir_name_chars
                             <1>
                             <1>       ;mov   [DelFile_FNPointer], esi
                             <1>       ;cmp   byte [esi], 20h
                             <1>       ;ja    short loc_rename_df_find
                             <1>
                             <1>       ;mov   dh, [Current_Drv] ; dh has not been changed
                             <1>
                             <1> rename_df_drv_check_writable:
00008F84 0FB6F6              <1>       movzx  esi, dh
                             <1>       ;movzx esi, byte [Current_Drv]
00008F87 81C600010900        <1>       add    esi, Logical_DOSDisks
                             <1>
00008F8D 88F2                <1>       mov    dl, dh ; dl = [Current_Drv]
00008F8F 8A7601              <1>       mov    dh, [esi+LD_DiskType]
                             <1>
00008F92 80FE01              <1>       cmp    dh, 1 ; 0 = Invalid
00008F95 7310                <1>       jnb    short rename_df_compare_sf_df_name
                             <1>
                             <1>       ; 16/10/2016 (13h -> 30)
00008F97 B81E000000          <1>       mov    eax, 30 ; 'Disk write-protected' error
00008F9C 8B1D[E4630100]      <1>       mov    ebx, [DestinationFilePath]
00008FA2 E9EAF6FFFF          <1>       jmp    loc_file_rw_cmd_failed
                             <1>
                             <1> rename_df_compare_sf_df_name:
00008FA7 BE[E4620100]        <1>       mov    esi, FindFile_Name
00008FAC BF[2A640100]        <1>       mov    edi, SourceFile_Name
00008FB1 B90C000000          <1>       mov    ecx, 12
                             <1> rename_df_compare_sf_df_name_next:
00008FB6 AC                  <1>       lodsb
00008FB7 AE                  <1>       scasb
00008FB8 7506                <1>       jne    short loc_rename_df_find
00008FBA 08C0                <1>       or     al, al
00008FBC 74AB                <1>       jz     short loc_rename_df_cmd_failed
00008FBE E2F6                <1>       loop   rename_df_compare_sf_df_name_next
                             <1>
                             <1> loc_rename_df_find:
                             <1>       ;mov   esi, [DelFile_FNPointer]
00008FC0 BE[E4620100]        <1>       mov    esi, FindFile_Name
                             <1>
00008FC5 6631C0              <1>       xor    ax, ax ; Any
00008FC8 E896F2FFFF          <1>       call   find_first_file
                             <1>       ;jnc   short loc_rename_df_found
                             <1>       ; 29/12/2017
00008FCD 0F83C8F6FFFF        <1>       jnc    loc_permission_denied
                             <1>
                             <1> loc_rename_df_check_error_code:
                             <1>       ;cmp   eax, 2
00008FD3 3C02                <1>       cmp    al, 2 ; Not found error
00008FD5 7406                <1>       je     short rename_df_move_find_struct_to_dest
00008FD7 F9                  <1>       stc
00008FD8 E9B4F6FFFF          <1>       jmp    loc_file_rw_cmd_failed
                             <1>
                             <1> ;loc_rename_df_found:
                             <1>       ; 05/11/2016
                             <1>       ; Permission denied error
                             <1>       ;mov   eax, ERR_PERM_DENIED ; 29/12/2017
                             <1>       ;stc
                             <1>       ;jmp   loc_permission_denied  ; 06/11/2016
                             <1>
                             <1> rename_df_move_find_struct_to_dest:
00008FDD BE[A2620100]        <1>         mov     esi, FindFile_Drv
00008FE2 BF[68640100]        <1>         mov     edi, DestinationFile_Drv
00008FE7 B920000000          <1>       mov    ecx, 32
```

```
00008FEC F3A5                <1>       rep    movsd
                             <1>
                             <1> loc_rename_df_process_q_sf:
                             <1>       ;mov   ecx, 12
00008FEE B10C                <1>       mov    cl, 12
00008FF0 BE[2A640100]        <1>       mov    esi, SourceFile_Name
00008FF5 BF[23120100]        <1>       mov    edi, Rename_OldName
                             <1> rename_df_process_q_nml_1_sf:
00008FFA AC                  <1>       lodsb
00008FFB 3C20                <1>         cmp al, 20h
00008FFD 7603                <1>         jna short rename_df_process_q_nml_2_sf
00008FFF AA                  <1>       stosb
00009000 E2F8                <1>       loop   rename_df_process_q_nml_1_sf
                             <1>
                             <1> rename_df_process_q_nml_2_sf:
00009002 C60700              <1>       mov    byte [edi], 0
                             <1>
                             <1> loc_rename_df_process_q_df:
                             <1>       ;mov   ecx, 12
00009005 B10C                <1>       mov    cl, 12
00009007 BE[AA640100]        <1>       mov    esi, DestinationFile_Name
0000900C BF[34120100]        <1>       mov    edi, Rename_NewName
                             <1> rename_df_process_q_nml_1_df:
00009011 AC                  <1>       lodsb
00009012 3C20                <1>       cmp    al, 20h
00009014 7603                <1>       jna    short loc_rename_df_process_q_nml_2_df
00009016 AA                  <1>       stosb
00009017 E2F8                <1>       loop   rename_df_process_q_nml_1_df
                             <1>
                             <1> loc_rename_df_process_q_nml_2_df:
00009019 C60700              <1>       mov    byte [edi], 0
                             <1>
                             <1> loc_rename_confirmation_question:
0000901C BE[FB110100]        <1>       mov    esi, Msg_DoYouWantRename
00009021 E837D3FFFF          <1>       call   print_msg
                             <1>
00009026 A0[45640100]        <1>       mov    al, [SourceFile_DirEntry+11] ; Attributes
0000902B 2410                <1>       and    al, 10h
0000902D 750C                <1>       jnz    short rename_confirmation_question_dir
                             <1>
                             <1> rename_confirmation_question_file:
0000902F BE[12120100]        <1>       mov    esi, Rename_File
00009034 E824D3FFFF          <1>       call   print_msg
00009039 EB0A                <1>       jmp    short rename_confirmation_question_as
                             <1>
                             <1> rename_confirmation_question_dir:
0000903B BE[18120100]        <1>       mov    esi, Rename_Directory
00009040 E818D3FFFF          <1>       call   print_msg
                             <1>
                             <1> rename_confirmation_question_as:
00009045 BE[23120100]        <1>       mov    esi, Rename_OldName
0000904A E80ED3FFFF          <1>       call   print_msg
0000904F BE[30120100]        <1>       mov    esi, Msg_File_rename_as
00009054 E804D3FFFF          <1>       call   print_msg
00009059 BE[57110100]        <1>       mov    esi, Msg_YesNo
0000905E E8FAD2FFFF          <1>       call   print_msg
                             <1>
                             <1> loc_rename_ask_again:
00009063 30E4                <1>       xor    ah, ah
00009065 E8AC7BFFFF          <1>       call   int16h
0000906A 3C1B                <1>       cmp    al, 1Bh
0000906C 740F                <1>       je     short loc_do_not_rename_file
0000906E 24DF                <1>       and    al, 0DFh
00009070 A2[61110100]        <1>       mov    [Y_N_nextline], al
00009075 3C59                <1>       cmp    al, 'Y'
00009077 7404                <1>       je     short loc_yes_rename_file
00009079 3C4E                <1>       cmp    al, 'N'
0000907B 75E6                <1>       jne    short loc_rename_ask_again
                             <1>
                             <1> loc_do_not_rename_file:
                             <1> loc_yes_rename_file:
0000907D E848F7FFFF          <1>       call   y_n_answer ; 29/12/2017
                             <1>       ;cmp   al, 'Y' ; 'yes'
                             <1>       ;cmc
                             <1>         ;jnc loc_file_rw_restore_retn
00009082 3C4E                <1>       cmp    al, 'N' ; 'no'
00009084 0F8407F6FFFF        <1>         je   loc_file_rw_restore_retn
                             <1>
0000908A BE[34120100]        <1>       mov    esi, Rename_NewName
0000908F 668B0D[62640100]    <1>       mov    cx, [SourceFile_DirEntryNumber]
00009096 66A1[4E640100]      <1>       mov    ax, [SourceFile_DirEntry+20] ; First Cluster, HW
0000909C C1E010              <1>       shl    eax, 16 ; 13/11/2017
0000909F 66A1[54640100]      <1>       mov    ax, [SourceFile_DirEntry+26] ; First Cluster, LW
                             <1>
000090A5 0FB61D[37640100]    <1>       movzx  ebx, byte [SourceFile_LongNameEntryLength]
000090AC E8CB1B0000          <1>       call   rename_directory_entry
000090B1 E9FBF6FFFF          <1>       jmp    loc_rename_file_ok
                             <1> ;loc_rename_file_ok:
                             <1> ;     jc     loc_run_cmd_failed
                             <1> ;     mov    esi, Msg_OK
                             <1> ;     call   proc_printmsg
                             <1> ;     jmp    loc_file_rw_restore_retn
                             <1>
                             <1> move_file:
                             <1>       ; 11/03/2016
                             <1>       ; 09/03/2016
                             <1>       ; 08/03/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 21/05/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_move')
                             <1>       ; 23/04/2011
                             <1>
                             <1> get_move_source_fchar:
                             <1>       ; esi = file name
000090B6 803E20              <1>       cmp    byte [esi], 20h
```

```
000090B9 7614                <1>         jna  short loc_move_nofilename_retn
                             <1> 
000090BB 8935[E0630100]      <1>       mov    [SourceFilePath], esi
                             <1> 
                             <1> move_scan_source_file:
000090C1 46                  <1>       inc    esi
000090C2 803E20              <1>       cmp    byte [esi], 20h
000090C5 7409                <1>         je      short move_scan_destination_1
                             <1>       ;jb    short loc_move_nofilename_retn
000090C7 0F8286ECFFFF        <1>       jb     loc_cmd_failed
000090CD EBF2                <1>       jmp    short move_scan_source_file
                             <1> 
                             <1> loc_move_nofilename_retn:
000090CF C3                  <1>       retn
                             <1> 
                             <1> move_scan_destination_1:
000090D0 C60600              <1>       mov    byte [esi], 0
                             <1> 
                             <1> move_scan_destination_2:
000090D3 46                  <1>       inc    esi
000090D4 803E20              <1>       cmp    byte [esi], 20h
000090D7 74FA                <1>       je     short move_scan_destination_2
                             <1>       ;jb    short loc_move_nofilename_retn
000090D9 0F8274ECFFFF        <1>       jb     loc_cmd_failed
                             <1> 
000090DF 8935[E4630100]      <1>       mov    [DestinationFilePath], esi
                             <1> 
                             <1> move_scan_destination_3:
000090E5 46                  <1>       inc    esi
000090E6 803E20              <1>       cmp    byte [esi], 20h
000090E9 77FA                <1>       ja     short move_scan_destination_3
000090EB C60600              <1>       mov    byte [esi], 0
                             <1> 
                             <1> loc_move_scan_destination_OK:
000090EE 8B35[E0630100]      <1>       mov    esi, [SourceFilePath]
000090F4 8B3D[E4630100]      <1>       mov    edi, [DestinationFilePath]
                             <1> 
000090FA B001                <1>       mov    al, 1  ; move procedure Phase 1
000090FC E8F71B0000          <1>       call   move_source_file_to_destination_file
00009101 7328                <1>       jnc    short move_source_file_to_destination_question
                             <1> 
                             <1> loc_move_cmd_failed_1:
00009103 08C0                <1>       or     al, al
00009105 0F8448ECFFFF        <1>       jz     loc_cmd_failed
0000910B 3C11                <1>       cmp    al, 11h
0000910D 740D                <1>       je     short loc_msg_not_same_device
                             <1>       ;cmp   al, 05h
                             <1>       ;cmp   al, ERR_PERM_DENIED ; 29/12/2017
                             <1>       ;jne   loc_run_cmd_failed
                             <1>       ;jmp   loc_permission_denied
0000910F 3C0B                <1>       cmp    al, ERR_PERM_DENIED
00009111 0F8484F5FFFF        <1>       je     loc_permission_denied
00009117 E962ECFFFF          <1>       jmp    loc_run_cmd_failed
                             <1> 
                             <1>       ;mov   esi, Msg_Permission_denied
                             <1>       ;call  print_msg
                             <1>       ;jmp   loc_file_rw_restore_retn
                             <1> 
                             <1> loc_msg_not_same_device:
0000911C BE[41120100]        <1>       mov    esi, msg_not_same_drv
00009121 E837D2FFFF          <1>       call   print_msg
00009126 E966F5FFFF          <1>       jmp    loc_file_rw_restore_retn
                             <1> 
                             <1> move_source_file_to_destination_question:
0000912B A0[E8630100]        <1>         mov     al, [SourceFile_Drv]
00009130 0441                <1>       add    al, 'A'
00009132 A2[A3120100]        <1>       mov    [msg_source_file_drv], al
00009137 A0[68640100]        <1>         mov     al, [DestinationFile_Drv]
0000913C 0441                <1>       add    al, 'A'
0000913E A2[C2120100]        <1>       mov    [msg_destination_file_drv], al
                             <1> 
00009143 57                  <1>       push   edi ; *
                             <1> 
00009144 BE[87120100]        <1>       mov    esi, msg_source_file
00009149 E80FD2FFFF          <1>       call   print_msg
0000914E BE[E9630100]        <1>       mov    esi, SourceFile_Directory
00009153 803E20              <1>       cmp    byte [esi], 20h
00009156 7605                <1>       jna    short msftdfq_sfn
00009158 E800D2FFFF          <1>       call   print_msg
                             <1> msftdfq_sfn:
0000915D BE[2A640100]        <1>       mov    esi, SourceFile_Name
00009162 E8F6D1FFFF          <1>       call   print_msg
00009167 BE[A6120100]        <1>       mov    esi, msg_destination_file
0000916C E8ECD1FFFF          <1>       call   print_msg
00009171 BE[69640100]        <1>       mov    esi, DestinationFile_Directory
00009176 803E20              <1>       cmp    byte [esi], 20h
00009179 7605                <1>       jna    short msftdfq_dfn
0000917B E8DDD1FFFF          <1>       call   print_msg
                             <1> msftdfq_dfn:
00009180 BE[AA640100]        <1>       mov    esi, DestinationFile_Name
00009185 E8D3D1FFFF          <1>       call   print_msg
0000918A BE[C5120100]        <1>       mov    esi, msg_copy_nextline
0000918F E8C9D1FFFF          <1>       call   print_msg
00009194 BE[C5120100]        <1>       mov    esi, msg_copy_nextline
00009199 E8BFD1FFFF          <1>       call   print_msg
                             <1> 
                             <1> loc_move_ask_for_new_file_yes_no:
0000919E BE[53120100]        <1>       mov    esi, Msg_DoYouWantMoveFile
000091A3 E8B5D1FFFF          <1>       call   print_msg
000091A8 BE[57110100]        <1>       mov    esi, Msg_YesNo
000091AD E8ABD1FFFF          <1>       call   print_msg
                             <1> loc_move_ask_for_new_file_again:
000091B2 30E4                <1>       xor    ah, ah
```

```
000091B4 E85D7AFFFF           <1>       call   int16h
000091B9 3C1B                 <1>       cmp    al, 1Bh
                              <1>       ;je    short loc_do_not_move_file
000091BB 7441                 <1>       je     short loc_move_y_n_escape
000091BD 24DF                 <1>       and    al, 0DFh
000091BF A2[61110100]         <1>         mov     [Y_N_nextline], al
000091C4 3C59                 <1>       cmp    al, 'Y'
000091C6 7404                 <1>       je     short loc_yes_move_file
000091C8 3C4E                 <1>       cmp    al, 'N'
000091CA 75E6                 <1>       jne    short loc_move_ask_for_new_file_again
                              <1> 
                              <1> loc_do_not_move_file:
                              <1> loc_yes_move_file:
000091CC E8F9F5FFFF           <1>       call   y_n_answer ; 29/12/2017
000091D1 5F                   <1>       pop    edi ; *
                              <1>       ;cmp   al, 'Y' ; 'yes'
                              <1>       ;cmc
                              <1>         ;jnc loc_file_rw_restore_retn
000091D2 3C4E                 <1>       cmp    al, 'N' ; 'no'
000091D4 0F84B7F4FFFF         <1>         je   loc_file_rw_restore_retn
                              <1> 
                              <1> loc_move_yes_move_file:
000091DA B002                 <1>       mov    al, 2 ; move procedure Phase 2
000091DC E8171B0000           <1>       call   move_source_file_to_destination_file
                              <1>       ;jc    short loc_move_cmd_failed_2
000091E1 0F83D0F5FFFF         <1>         jnc     move_source_file_to_destination_OK
                              <1> 
                              <1> ;move_source_file_to_destination_OK:
                              <1> ;     mov    esi, Msg_OK
                              <1> ;     call   print_msg
                              <1> ;     jmp    loc_file_rw_restore_retn
                              <1> 
                              <1> loc_move_cmd_failed_2:
000091E7 3C27                 <1>       cmp    al, 27h
000091E9 0F858FEBFFFF         <1>       jne    loc_run_cmd_failed
                              <1> 
000091EF BE[6C120100]         <1>       mov    esi, msg_insufficient_disk_space
000091F4 E864D1FFFF           <1>       call   print_msg
                              <1> 
000091F9 E993F4FFFF           <1>       jmp    loc_file_rw_restore_retn
                              <1> 
                              <1> loc_move_y_n_escape:
000091FE B04E                 <1>       mov    al, 'N' ; 'no'
00009200 EBCA                 <1>       jmp    short loc_do_not_move_file
                              <1> 
                              <1> copy_file:
                              <1>       ; 15/10/2016
                              <1>       ; 24/03/2016
                              <1>       ; 21/03/2016
                              <1>       ; 15/03/2016 (TRDOS 386 = TRDOS v2.0)
                              <1>       ; 21/05/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_copy')
                              <1>       ; 01/08/2010
                              <1> 
                              <1> get_copy_source_fchar:
                              <1>       ; esi = file name
00009202 803E20               <1>       cmp    byte [esi], 20h
00009205 7614                 <1>         jna     short loc_copy_nofilename_retn
                              <1> 
00009207 8935[E0630100]       <1>       mov    [SourceFilePath], esi
                              <1> 
                              <1> copy_scan_source_file:
0000920D 46                   <1>       inc    esi
0000920E 803E20               <1>       cmp    byte [esi], 20h
00009211 7409                 <1>       je     short copy_scan_destination_1
                              <1>       ;jb    short loc_copy_nofilename_retn
00009213 0F823AEBFFFF         <1>       jb     loc_cmd_failed
00009219 EBF2                 <1>       jmp    short copy_scan_source_file
                              <1> 
                              <1> loc_copy_nofilename_retn:
0000921B C3                   <1>       retn
                              <1> 
                              <1> copy_scan_destination_1:
0000921C C60600               <1>       mov    byte [esi], 0
                              <1> 
                              <1> copy_scan_destination_2:
0000921F 46                   <1>       inc    esi
00009220 803E20               <1>       cmp    byte [esi], 20h
00009223 74FA                 <1>       je     short copy_scan_destination_2
                              <1>       ;jb    short loc_copy_nofilename_retn
00009225 0F8228EBFFFF         <1>       jb     loc_cmd_failed
                              <1> 
0000922B 8935[E4630100]       <1>       mov    [DestinationFilePath], esi
                              <1> 
                              <1> copy_scan_destination_3:
00009231 46                   <1>       inc    esi
00009232 803E20               <1>       cmp    byte [esi], 20h
00009235 77FA                 <1>       ja     short copy_scan_destination_3
00009237 C60600               <1>       mov    byte [esi], 0
                              <1> 
                              <1> loc_copy_save_current_drive:
0000923A 8A35[FE580100]       <1>       mov    dh, [Current_Drv]
00009240 8835[5E610100]       <1>       mov    [RUN_CDRV], dh
                              <1> 
                              <1> copy_source_file_to_destination_phase_1:
00009246 8B35[E0630100]       <1>       mov    esi, [SourceFilePath]
0000924C 8B3D[E4630100]       <1>       mov    edi, [DestinationFilePath]
                              <1> 
00009252 B001                 <1>       mov    al, 1  ; copy procedure Phase 1
00009254 E83C1D0000           <1>       call   copy_source_file_to_destination_file
00009259 732B                 <1>       jnc    short copy_source_file_to_destination_question
                              <1> 
                              <1> loc_copy_cmd_failed_1:
                              <1>       ; 18/03/2016 (restore current drive and directory)
```

```
0000925B 08C0                <1>       or     al, al
0000925D 7507                <1>       jnz    short loc_copy_cmd_failed_2
                             <1> 
0000925F FEC0                <1>         inc     al ; mov al, 1 ; Bad command or file name !
00009261 E918EBFFFF          <1>       jmp    loc_run_cmd_failed
                             <1> 
                             <1> loc_copy_cmd_failed_2:
00009266 3C27                <1>       cmp    al, 27h ; Insufficient disk space
00009268 740D                <1>       je     short loc_file_write_insuff_disk_space_msg
                             <1> 
                             <1>       ; 29/12/2017
                             <1>       ;cmp   al, 05h
0000926A 3C0B                <1>       cmp    al, ERR_PERM_DENIED
0000926C 0F850CEBFFFF        <1>       jne    loc_run_cmd_failed
                             <1> 
00009272 E924F4FFFF          <1>       jmp    loc_permission_denied
                             <1> 
                             <1> loc_file_write_insuff_disk_space_msg:
00009277 BE[6C120100]        <1>       mov    esi, msg_insufficient_disk_space
0000927C E8DCD0FFFF          <1>       call   print_msg
00009281 E90BF4FFFF          <1>         jmp     loc_file_rw_restore_retn
                             <1> 
                             <1> copy_source_file_to_destination_question:
00009286 57                  <1>       push   edi ; *
                             <1> 
                             <1>       ; dh = source file attributes
                             <1>       ; dl > 0 -> destination file found
00009287 20D2                <1>       and    dl, dl
00009289 7449                <1>       jz     short copy_source_file_to_destination_pass_owrq
                             <1> 
                             <1> loc_copy_ask_for_owr_yes_no:
0000928B BE[C8120100]        <1>       mov    esi, Msg_DoYouWantOverWriteFile
00009290 E8C8D0FFFF          <1>       call   print_msg
00009295 BE[AA640100]        <1>       mov    esi, DestinationFile_Name
0000929A E8BED0FFFF          <1>       call   print_msg
0000929F BE[57110100]        <1>       mov    esi, Msg_YesNo
000092A4 E8B4D0FFFF          <1>       call   print_msg
                             <1> 
                             <1> loc_copy_ask_for_owr_again:
000092A9 30E4                <1>       xor    ah, ah
000092AB E86679FFFF          <1>       call   int16h
000092B0 3C1B                <1>       cmp    al, 1Bh
                             <1>         ;je     loc_do_not_copy_file
000092B2 7419                <1>         je      short loc_copy_y_n_escape
000092B4 24DF                <1>       and    al, 0DFh
000092B6 A2[61110100]        <1>         mov     [Y_N_nextline], al
000092BB 3C59                <1>       cmp    al, 'Y'
000092BD 0F84B1000000        <1>         je      loc_yes_copy_file
000092C3 3C4E                <1>       cmp    al, 'N'
000092C5 0F84A9000000        <1>         je      loc_do_not_copy_file
000092CB EBDC                <1>       jmp    short loc_copy_ask_for_owr_again
                             <1> 
                             <1> loc_copy_y_n_escape:
000092CD B04E                <1>       mov    al, 'N' ; 'no'
000092CF E9A0000000          <1>         jmp     loc_do_not_copy_file
                             <1> 
                             <1> copy_source_file_to_destination_pass_owrq:
000092D4 A0[E8630100]        <1>       mov     al, [SourceFile_Drv]
000092D9 0441                <1>       add    al, 'A'
000092DB A2[A3120100]        <1>       mov    [msg_source_file_drv], al
000092E0 A0[68640100]        <1>         mov     al, [DestinationFile_Drv]
000092E5 0441                <1>       add    al, 'A'
000092E7 A2[C2120100]        <1>       mov    [msg_destination_file_drv], al
                             <1> 
000092EC BE[87120100]        <1>       mov    esi, msg_source_file
000092F1 E867D0FFFF          <1>       call   print_msg
000092F6 BE[E9630100]        <1>       mov    esi, SourceFile_Directory
000092FB 803E20              <1>       cmp    byte [esi], 20h
000092FE 7605                <1>       jna    short csftdfq_sfn
00009300 E858D0FFFF          <1>       call   print_msg
                             <1> csftdfq_sfn:
00009305 BE[2A640100]        <1>       mov    esi, SourceFile_Name
0000930A E84ED0FFFF          <1>       call   print_msg
0000930F BE[A6120100]        <1>       mov    esi, msg_destination_file
00009314 E844D0FFFF          <1>       call   print_msg
00009319 BE[69640100]        <1>       mov    esi, DestinationFile_Directory
0000931E 803E20              <1>       cmp    byte [esi], 20h
00009321 7605                <1>       jna    short csftdfq_dfn
00009323 E835D0FFFF          <1>       call   print_msg
                             <1> csftdfq_dfn:
00009328 BE[AA640100]        <1>       mov    esi, DestinationFile_Name
0000932D E82BD0FFFF          <1>       call   print_msg
00009332 BE[C5120100]        <1>       mov    esi, msg_copy_nextline
00009337 E821D0FFFF          <1>       call   print_msg
0000933C BE[C5120100]        <1>       mov    esi, msg_copy_nextline
00009341 E817D0FFFF          <1>       call   print_msg
                             <1> 
                             <1> loc_copy_ask_for_new_file_yes_no:
00009346 BE[E7120100]        <1>       mov    esi, Msg_DoYouWantCopyFile
0000934B E80DD0FFFF          <1>       call   print_msg
00009350 BE[57110100]        <1>       mov    esi, Msg_YesNo
00009355 E803D0FFFF          <1>       call   print_msg
                             <1> 
                             <1> loc_copy_ask_for_new_file_again:
0000935A 30E4                <1>       xor    ah, ah
0000935C E8B578FFFF          <1>       call   int16h
00009361 3C1B                <1>       cmp    al, 1Bh
00009363 740F                <1>       je     short loc_do_not_copy_file
00009365 24DF                <1>       and    al, 0DFh
00009367 A2[61110100]        <1>         mov     [Y_N_nextline], al
0000936C 3C59                <1>       cmp    al, 'Y'
0000936E 7404                <1>       je     short loc_yes_copy_file
00009370 3C4E                <1>       cmp    al, 'N'
```

```
00009372 75E6                <1>       jne    short loc_copy_ask_for_new_file_again
                             <1> 
                             <1> loc_do_not_copy_file:
                             <1> loc_yes_copy_file:
00009374 E851F4FFFF          <1>       call   y_n_answer ; 29/12/2017
00009379 5F                  <1>       pop    edi ; *
                             <1>       ;cmp   al, 'Y' ; 'yes'
                             <1>       ;cmc
                             <1>         ;jnc loc_file_rw_restore_retn
0000937A 3C4E                <1>       cmp    al, 'N' ; 'no'
0000937C 0F840FF3FFFF        <1>         je   loc_file_rw_restore_retn
                             <1> 
                             <1> copy_source_file_to_destination_pass_q:
00009382 B002                <1>       mov    al, 2  ; copy procedure Phase 2
00009384 E80C1C0000          <1>       call   copy_source_file_to_destination_file
                             <1>       ;jc    short loc_file_write_check_disk_space_err
                             <1> 
                             <1>       ; 24/03/2016
                             <1>       ;push  cx
00009389 51                  <1>       push   ecx ; 29/12/2017
0000938A BE[C5120100]        <1>       mov    esi, msg_copy_nextline
0000938F E8C9CFFFFF          <1>       call   print_msg
00009394 58                  <1>       pop    eax ; 29/12/2017
                             <1>       ;;pop  cx
                             <1>       ;pop   ax
                             <1> 
                             <1>       ;or    cl, cl
00009395 08C0                <1>       or     al, al
00009397 7419                <1>       jz     short copy_source_file_to_destination_OK
                             <1> 
                             <1>       ; 15/10/2016 (1Dh -> 18)
                             <1>       ; 18/03/2016 (1Dh)
                             <1>       ;cmp   cl, 18 ; write error
00009399 3C12                <1>       cmp    al, 18
0000939B 7506                <1>       jne    short copy_source_file_to_destination_not_OK
                             <1>       ;
                             <1>       ;mov   al, cl ; error number (write fault!)
0000939D F9                  <1>       stc
0000939E E9EEF2FFFF          <1>       jmp    loc_file_rw_cmd_failed
                             <1> 
                             <1> copy_source_file_to_destination_not_OK:
000093A3 BE[00130100]        <1>       mov    esi, Msg_read_file_error_before_EOF
000093A8 E8B0CFFFFF          <1>       call   print_msg
000093AD E9DFF2FFFF          <1>       jmp    loc_file_rw_restore_retn
                             <1> 
                             <1> copy_source_file_to_destination_OK:
000093B2 BE[65110100]        <1>       mov    esi, Msg_OK
000093B7 E8A1CFFFFF          <1>       call   print_msg
                             <1> 
000093BC E9D0F2FFFF          <1>       jmp    loc_file_rw_restore_retn
                             <1> 
                             <1> ;loc_file_write_check_disk_space_err:
                             <1>       ;cmp   al, 27h ; Insufficient disk space
                             <1>       ;je    loc_file_write_insuff_disk_space_msg
                             <1>         ;jb  loc_file_rw_cmd_failed
                             <1> 
                             <1>       ;call  print_misc_error_msg ; 15/03/2016
                             <1>         ;jmp loc_file_rw_restore_retn
                             <1> 
                             <1> change_fs_file_attributes:
                             <1>       ; 04/03/2016 ; Temporary
                             <1>       ; AL = File or directory attributes
                             <1>       ; AH = 0 -> Attributes are in MS-DOS format
                             <1>       ; AH > 0 -> Attributes are in SINGLIX format
                             <1>       ;push  ebx
                             <1>       ; ... do somethings here ...
                             <1>       ;pop   ebx
                             <1>       ; BL = File or directory attributes
000093C1 C3                  <1>       retn
                             <1> 
                             <1> set_get_env:
                             <1>       ; 11/04/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 02/09/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_set')
                             <1>       ; 2005 - 28/08/2011
                             <1> get_setenv_fchar:
                             <1>       ; esi = environment variable/string
000093C2 8A06                <1>       mov    al, [esi]
000093C4 3C20                <1>       cmp    al, 20h
000093C6 771E                <1>       ja     short loc_find_env
                             <1> 
000093C8 BE00300900          <1>       mov    esi, Env_Page
                             <1> loc_print_setline:
000093CD 803E00              <1>       cmp    byte [esi], 0
000093D0 7613                <1>       jna    short loc_setenv_retn
000093D2 E886CFFFFF          <1>       call   print_msg
000093D7 56                  <1>       push   esi
000093D8 BE[6F190100]        <1>       mov    esi, nextline
000093DD E87BCFFFFF          <1>       call   print_msg
000093E2 5E                  <1>       pop    esi
000093E3 EBE8                <1>       jmp    short loc_print_setline
                             <1> 
                             <1> loc_setenv_retn:
000093E5 C3                  <1>       retn
                             <1> 
                             <1> loc_find_env:
000093E6 3C3D                <1>       cmp    al, '='
000093E8 0F8465E9FFFF        <1>       je     loc_cmd_failed
                             <1> 
000093EE 56                  <1>       push   esi
                             <1> loc_repeat_env_equal_check:
000093EF 46                  <1>       inc    esi
000093F0 803E3D              <1>       cmp    byte [esi], '='
000093F3 7431                <1>       je     short pass_env_equal_check
```

```
000093F5 803E20              <1>       cmp    byte [esi], 20h
000093F8 73F5                <1>       jnb    short loc_repeat_env_equal_check
000093FA C60600              <1>       mov    byte [esi], 0
000093FD 5E                  <1>       pop    esi
000093FE BF[FE590100]        <1>       mov    edi, TextBuffer ; out buffer
00009403 B9FF000000          <1>       mov    ecx, 255 ; maximum size (limit)
00009408 30C0                <1>       xor    al, al ; 0 -> use [ESI]
0000940A E89E000000          <1>       call   get_environment_string
0000940F 72D4                <1>       jc     short loc_setenv_retn
                             <1>
00009411 BE[FE590100]        <1>       mov    esi, TextBuffer
00009416 E842CFFFFF          <1>       call   print_msg
0000941B BE[6F190100]        <1>       mov    esi, nextline
00009420 E838CFFFFF          <1>       call   print_msg
                             <1>
00009425 C3                  <1>       retn
                             <1>
                             <1> pass_env_equal_check:
00009426 46                  <1>       inc    esi
00009427 803E20              <1>       cmp    byte [esi], 20h
0000942A 73FA                <1>       jnb    short pass_env_equal_check
0000942C C60600              <1>       mov    byte [esi], 0
                             <1>
                             <1> loc_call_set_env_string:
0000942F 5E                  <1>       pop    esi
00009430 E83B010000          <1>       call   set_environment_string
00009435 73AE                <1>       jnc    short loc_setenv_retn
                             <1>
                             <1> loc_set_cmd_failed:
00009437 3C08                <1>       cmp    al, 08h
00009439 0F8514E9FFFF        <1>       jne    loc_cmd_failed
                             <1>
0000943F BE[40130100]        <1>       mov    esi, Msg_No_Set_Space
00009444 E814CFFFFF          <1>       call   print_msg
                             <1>
00009449 C3                  <1>       retn
                             <1>
                             <1> set_get_path:
                             <1>       ; 11/04/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 03/09/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_path')
                             <1>       ; 2005
                             <1> get_path_fchar:
                             <1>       ; esi = path
0000944A 803E20              <1>       cmp    byte [esi], 20h
0000944D 7737                <1>       ja     short loc_set_path
                             <1>
0000944F BE00300900          <1>       mov    esi, Env_Page
                             <1> loc_print_path:
00009454 803E00              <1>       cmp    byte [esi], 0
00009457 762C                <1>       jna    short loc_path_retn
                             <1>
00009459 BE[9F0D0100]        <1>       mov    esi, Cmd_Path ; 'PATH' address
0000945E BF[FE590100]        <1>       mov    edi, TextBuffer ; out buffer
00009463 30C0                <1>       xor    al, al  ; use [ESI]
00009465 B9FF000000          <1>       mov    ecx, 255 ; maximum size (limit)
0000946A E83E000000          <1>       call   get_environment_string
0000946F 7214                <1>       jc     short loc_path_retn
                             <1>
00009471 BE[FE590100]        <1>       mov    esi, TextBuffer
00009476 E8E2CEFFFF          <1>       call   print_msg
0000947B BE[6F190100]        <1>       mov    esi, nextline
00009480 E8D8CEFFFF          <1>       call   print_msg
                             <1>
                             <1> loc_path_retn:
00009485 C3                  <1>       retn
                             <1>
                             <1> loc_set_path:
00009486 56                  <1>       push   esi
                             <1> loc_set_path_find_end:
00009487 46                  <1>       inc    esi
00009488 803E20              <1>       cmp    byte [esi], 20h
0000948B 73FA                <1>       jnb    short loc_set_path_find_end
0000948D C60600              <1>       mov    byte [esi], 0
                             <1> loc_set_path_header:
00009490 5E                  <1>       pop    esi
                             <1> set_path_x: ; 31/12/2017 ('syspath')
00009491 4E                  <1>       dec    esi
00009492 C6063D              <1>       mov    byte [esi], '='
00009495 4E                  <1>       dec    esi
00009496 C60648              <1>       mov    byte [esi], 'H'
00009499 4E                  <1>       dec    esi
0000949A C60654              <1>       mov    byte [esi], 'T'
0000949D 4E                  <1>       dec    esi
0000949E C60641              <1>       mov    byte [esi], 'A'
000094A1 4E                  <1>       dec    esi
000094A2 C60650              <1>       mov    byte [esi], 'P'
                             <1>
                             <1> loc_path_call_set_env_string:
000094A5 E8C6000000          <1>       call   set_environment_string
000094AA 728B                <1>         jc   short loc_set_cmd_failed
                             <1>
000094AC C3                  <1>       retn
                             <1>
                             <1> get_environment_string:
                             <1>       ; 12/04/2016
                             <1>       ; 11/04/2016
                             <1>       ; 05/04/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 02/09/2011 (TRDOS v1, MAINPROG.ASM)
                             <1>       ; 28/08/2011
                             <1>       ; INPUT->
                             <1>       ;      EDI = Output buffer
                             <1>       ;      CX = Buffer length (<= ENV_PAGE_SIZE)
                             <1>       ;
                             <1>       ;      AL > 0 = AL = String sequence number
                             <1>       ;      AL = 0 -> ESI = ASCIIZ Set word
                             <1>       ;            (environment variable)
                             <1>       ; OUTPUT ->
                             <1>       ;      ESI is not changed
                             <1>       ;      EDI is not changed
                             <1>       ;      EAX = String length (with zero tail)
                             <1>       ;      EDX = Environment variables page address
                             <1>       ;      CF = 1 -> Not found (EAX not valid)
                             <1>       ;
                             <1>       ; (Modified registers: EAX, EDX)
                             <1>
000094AD BA00300900          <1>       mov    edx, Env_Page
000094B2 803A00              <1>       cmp    byte [edx], 0
000094B5 7474                <1>       jz     short get_env_string_with_word_stc_retn
                             <1>
000094B7 66890D[6C650100]    <1>       mov    [env_var_length], cx
                             <1>
000094BE 51                  <1>       push   ecx ; *
000094BF 56                  <1>       push   esi ; **
                             <1>
000094C0 08C0                <1>       or     al, al
000094C2 7449                <1>       jz     short get_env_string_with_word
                             <1>
                             <1> get_env_string_with_seq_number:
000094C4 B101                <1>       mov    cl, 1
000094C6 88C5                <1>       mov    ch, al
000094C8 31C0                <1>       xor    eax, eax
000094CA 89D6                <1>       mov    esi, edx ; Env_Page
                             <1>
                             <1> get_env_string_seq_number_check:
000094CC 38CD                <1>       cmp    ch, cl
000094CE 7726                <1>       ja     short get_env_string_seq_number_next
                             <1>
                             <1> get_env_string_move_to_buff:
000094D0 57                  <1>       push   edi ; ***
                             <1>
000094D1 29D2                <1>       sub    edx, edx
                             <1>
                             <1> get_env_string_seq_number_repeat1:
000094D3 42                  <1>       inc    edx
000094D4 AC                  <1>       lodsb
000094D5 AA                  <1>       stosb
                             <1>
000094D6 66FF0D[6C650100]    <1>       dec    word [env_var_length]
000094DD 7508                <1>       jnz    short get_env_string_seq_number_repeat3
                             <1>
                             <1> get_env_string_seq_number_repeat2:
000094DF 20C0                <1>       and    al, al
000094E1 7408                <1>       jz     short get_env_string_seq_number_ok
000094E3 42                  <1>       inc    edx
000094E4 AC                  <1>       lodsb
000094E5 EBF8                <1>       jmp    short get_env_string_seq_number_repeat2
                             <1>
                             <1> get_env_string_seq_number_repeat3:
000094E7 08C0                <1>       or     al, al
000094E9 75E8                <1>       jnz    short get_env_string_seq_number_repeat1
                             <1>
                             <1> get_env_string_seq_number_ok:
000094EB 5F                  <1>       pop    edi ; ***
000094EC 89D0                <1>       mov    eax, edx ; Length of the environment string
                             <1>              ; (ASCIIZ, includes ZERO tail)
000094EE BA00300900          <1>       mov    edx, Env_Page
                             <1>
                             <1> get_env_string_stc_retn:
000094F3 5E                  <1>       pop    esi ; **
000094F4 59                  <1>       pop    ecx ; *
000094F5 C3                  <1>       retn
                             <1>
                             <1> get_env_string_seq_number_next:
000094F6 AC                  <1>       lodsb
000094F7 08C0                <1>       or     al, al
000094F9 75FB                <1>       jnz    short get_env_string_seq_number_next
                             <1>
000094FB 81FE00320900        <1>       cmp    esi, Env_Page + Env_Page_Size ; +512 (+4096)
00009501 F5                  <1>       cmc
00009502 72EF                <1>       jc     short get_env_string_stc_retn
                             <1>
00009504 AC                  <1>       lodsb
00009505 3C01                <1>       cmp    al, 1
00009507 72EA                <1>       jb     short get_env_string_stc_retn
00009509 FEC1                <1>       inc    cl
0000950B EBBF                <1>       jmp    short get_env_string_seq_number_check
                             <1>
                             <1> get_env_string_with_word:
0000950D 31C9                <1>       xor    ecx, ecx
                             <1>
                             <1> get_env_string_calc_word_length:
0000950F AC                  <1>       lodsb
00009510 3C20                <1>       cmp    al, 20h
00009512 7211                <1>       jb     short get_env_string_calc_word_length_ok
                             <1>       ;inc   cx
00009514 FEC1                <1>       inc    cl
                             <1>
00009516 3C61                <1>       cmp    al, 'a'
00009518 72F5                <1>       jb     short get_env_string_calc_word_length
0000951A 3C7A                <1>       cmp    al, 'z'
0000951C 77F1                <1>       ja     short get_env_string_calc_word_length
0000951E 24DF                <1>       and    al, 0DFh
00009520 8846FF              <1>       mov    [esi-1], al
00009523 EBEA                <1>       jmp    short get_env_string_calc_word_length
                             <1>
                             <1> get_env_string_calc_word_length_ok:
```

```
00009525 08C9                 <1>       or     cl, cl
00009527 7506                 <1>       jnz    short get_env_string_calc_word_length_save
                              <1> 
00009529 5E                   <1>       pop    esi ; **
                              <1> 
                              <1> get_env_string_stc_retn1:
0000952A 59                   <1>       pop    ecx ; *
                              <1> 
                              <1> get_env_string_with_word_stc_retn:
0000952B 31C0                 <1>       xor    eax, eax
0000952D F9                   <1>       stc
0000952E C3                   <1>       retn
                              <1> 
                              <1> get_env_string_calc_word_length_save:
0000952F 871C24               <1>       xchg   ebx, [esp] ; **
00009532 89DE                 <1>       mov    esi, ebx
                              <1>              ; Start of the env string (to be searched)
                              <1> 
00009534 57                   <1>       push   edi ; ***
00009535 89D7                 <1>       mov    edi, edx ; Env_Page
                              <1> 
                              <1> get_env_string_compare:
00009537 57                   <1>       push   edi ; ****
00009538 51                   <1>       push   ecx ; ***** ; Variable name length
                              <1> 
                              <1> get_env_string_compare_rep:
00009539 AC                   <1>       lodsb
0000953A AE                   <1>       scasb
0000953B 7511                 <1>       jne    short get_env_string_compare_next1
0000953D E2FA                 <1>       loop   get_env_string_compare_rep
                              <1> 
0000953F 803F3D               <1>       cmp    byte [edi], '='
00009542 750A                 <1>       jne    short get_env_string_compare_next1
                              <1> 
00009544 59                   <1>       pop    ecx ; *****
00009545 5F                   <1>       pop    edi ; ****
00009546 89FE                 <1>       mov    esi, edi
00009548 5F                   <1>       pop    edi ; ***
00009549 871C24               <1>       xchg   ebx, [esp] ; **
0000954C EB82                 <1>       jmp    short get_env_string_move_to_buff
                              <1> 
                              <1> get_env_string_compare_next1:
0000954E 89FE                 <1>       mov    esi, edi
00009550 59                   <1>       pop    ecx ; *****
00009551 5F                   <1>       pop    edi ; ****
                              <1> get_env_string_compare_next2:
00009552 81FEFF310900         <1>       cmp    esi, Env_Page + Env_Page_Size - 1 ; +511 (+4095)
00009558 7310                 <1>       jnb    short get_env_string_compare_not_ok
0000955A 20C0                 <1>       and    al, al
0000955C AC                   <1>       lodsb
0000955D 75F3                 <1>       jnz    short get_env_string_compare_next2
0000955F 08C0                 <1>       or     al, al
00009561 7407                 <1>       jz     short get_env_string_compare_not_ok
00009563 4E                   <1>       dec    esi ; 12/04/2016
00009564 89F7                 <1>       mov    edi, esi
00009566 89DE                 <1>       mov    esi, ebx
00009568 EBCD                 <1>       jmp    short get_env_string_compare
                              <1> 
                              <1> get_env_string_compare_not_ok:
0000956A 5F                   <1>       pop    edi ; ***
0000956B 89DE                 <1>       mov    esi, ebx
0000956D 5B                   <1>       pop    ebx ; **
0000956E EBBA                 <1>       jmp    short get_env_string_stc_retn1
                              <1> 
                              <1> set_environment_string:
                              <1>       ; 13/04/2016
                              <1>       ; 12/04/2016
                              <1>       ; 11/04/2016
                              <1>       ; 06/04/2016
                              <1>       ; 05/04/2016 (TRDOS 386 = TRDOS v2.0)
                              <1>       ; 02/09/2011 (TRDOS v1, MAINPROG.ASM)
                              <1>       ; 29/08/2011
                              <1>       ; 29/08/2011
                              <1>       ; INPUT->
                              <1>       ;      ESI = ASCIIZ environment string
                              <1>       ; OUTPUT ->
                              <1>       ;      ESI is not changed
                              <1>       ;      CF = 1 -> Could not set,
                              <1>       ;           insufficient environment space
                              <1>       ;
                              <1>       ; (EAX, EDX will be changed)
                              <1>       ;
                              <1>       ;    (EAX = Start address of the env string if > 0)
                              <1>       ;    (EDX = Environment string length)
                              <1> 
00009570 56                   <1>       push   esi ; *
                              <1> 
00009571 31C0                 <1>       xor    eax, eax
                              <1> 
                              <1> set_env_chk_validation1:
00009573 FEC4                 <1>       inc    ah ; variable (string) length
00009575 AC                   <1>       lodsb
00009576 3C3D                 <1>       cmp    al, '='
00009578 7415                 <1>       je     short set_env_chk_validation2
0000957A 3C20                 <1>       cmp    al, 20h
0000957C 720F                 <1>       jb     short set_env_string_stc
                              <1> 
                              <1>       ; 06/04/2016
0000957E 3C61                 <1>       cmp    al, 'a'
00009580 72F1                 <1>       jb     short set_env_chk_validation1
00009582 3C7A                 <1>       cmp    al, 'z'
00009584 77ED                 <1>       ja     short set_env_chk_validation1
00009586 2C20                 <1>       sub    al, 'a'-'A'
```

```
00009588 8846FF              <1>       mov    [esi-1], al
0000958B EBE6                <1>       jmp    short set_env_chk_validation1
                             <1> 
                             <1> set_env_string_stc:
0000958D 5E                  <1>       pop    esi ; *
                             <1>       ;stc
0000958E C3                  <1>       retn
                             <1> 
                             <1> set_env_chk_validation2:
0000958F 51                  <1>       push   ecx ; **
00009590 53                  <1>       push   ebx ; ***
00009591 57                  <1>       push   edi ; ****
                             <1> 
                             <1>       ; 12/04/2016
00009592 8B5C240C            <1>       mov    ebx, [esp+12]
                             <1> 
                             <1> set_env_chk_validation2w:
00009596 89F7                <1>       mov    edi, esi
00009598 4F                  <1>       dec    edi
                             <1> 
00009599 807FFF20            <1>       cmp    byte [edi-1], 20h
0000959D 771A                <1>       ja     short set_env_chk_validation2z
                             <1> 
0000959F 56                  <1>       push   esi
000095A0 89FE                <1>       mov    esi, edi
000095A2 4E                  <1>       dec    esi
                             <1> 
                             <1> set_env_chk_validation2x:
000095A3 4E                  <1>       dec    esi
                             <1> 
000095A4 39DE                <1>       cmp    esi, ebx
000095A6 7207                <1>       jb     short set_env_chk_validation2y
                             <1> 
000095A8 4F                  <1>       dec    edi
                             <1> 
000095A9 8A06                <1>       mov    al, [esi]
000095AB 8807                <1>       mov    [edi], al
                             <1> 
000095AD EBF4                <1>       jmp    short set_env_chk_validation2x
                             <1> 
                             <1> set_env_chk_validation2y:
000095AF 5E                  <1>       pop    esi
                             <1> 
                             <1>       ;mov   byte [ebx], 20h
                             <1> 
000095B0 43                  <1>       inc    ebx
000095B1 895C240C            <1>       mov    [esp+12], ebx
                             <1> 
000095B5 FECC                <1>       dec    ah ; 13/04/2016
                             <1> 
000095B7 EBDD                <1>       jmp    short set_env_chk_validation2w
                             <1> 
                             <1> set_env_chk_validation2z:
000095B9 BA00300900          <1>       mov    edx, Env_Page
000095BE 89D7                <1>       mov    edi, edx
                             <1> 
                             <1> set_env_chk_validation3:
000095C0 AC                  <1>       lodsb
000095C1 3C20                <1>       cmp    al, 20h
000095C3 74FB                <1>       je     short set_env_chk_validation3
                             <1> 
000095C5 9C                  <1>       pushf
                             <1> 
                             <1>       ; 12/04/2016
                             <1> set_env_chk_validation3n:
000095C6 3C61                <1>       cmp    al, 'a'
000095C8 720C                <1>       jb     short set_env_chk_validation3c
000095CA 3C7A                <1>       cmp    al, 'z'
000095CC 7705                <1>       ja     short set_env_chk_validation3x
000095CE 2C20                <1>       sub    al, 'a'-'A'
000095D0 8846FF              <1>       mov    [esi-1], al
                             <1> 
                             <1> set_env_chk_validation3x:
000095D3 AC                  <1>       lodsb
000095D4 EBF0                <1>       jmp    short set_env_chk_validation3n
                             <1> 
                             <1> set_env_chk_validation3c:
000095D6 3C20                <1>       cmp    al, 20h
000095D8 73F9                <1>       jnb    short set_env_chk_validation3x
                             <1> 
000095DA 803F00              <1>       cmp    byte [edi], 0
000095DD 7731                <1>       ja     short set_env_chk_validation4
                             <1> 
000095DF 9D                  <1>       popf
000095E0 7228                <1>       jb     short set_env_string_nothing
                             <1> 
000095E2 B900020000          <1>       mov    ecx, Env_Page_Size ; 512 (4096)
                             <1> 
000095E7 89DE                <1>       mov    esi, ebx ; 12/04/2016
                             <1> 
                             <1> set_env_string_copy_to_envb:
000095E9 AC                  <1>       lodsb
000095EA 3C20                <1>       cmp    al, 20h
000095EC 720A                <1>       jb     short set_env_string_copy_to_envb_z
000095EE AA                  <1>       stosb
000095EF E2F8                <1>       loop   set_env_string_copy_to_envb
                             <1> 
                             <1>       ; 11/04/2016
000095F1 89D7                <1>       mov    edi, edx ; Env_Page
000095F3 B900020000          <1>       mov    ecx, Env_Page_Size
                             <1> 
                             <1> set_env_string_copy_to_envb_z:
000095F8 52                  <1>       push   edx  ; Start address of the variable
```

```
000095F9 BA00020000           <1>       mov    edx, Env_Page_Size
000095FE 29CA                 <1>       sub    edx, ecx ; variable (string) length
                              <1> 
00009600 28C0                 <1>       sub    al, al ; 0
00009602 F3AA                 <1>       rep    stosb ; clear remain bytes of the env page
                              <1> 
00009604 58                   <1>       pop    eax  ; Start address of the variable
                              <1> 
                              <1> set_env_string_allocate_envb_retn:  ; stc or clc return
00009605 5F                   <1>       pop    edi ; ****
00009606 5B                   <1>       pop    ebx ; ***
00009607 59                   <1>       pop    ecx ; **
00009608 5E                   <1>       pop    esi ; *
00009609 C3                   <1>       retn
                              <1> 
                              <1> set_env_string_nothing:
0000960A 31C0                 <1>       xor    eax, eax
0000960C 31D2                 <1>       xor    edx, edx ; 11/04/2016
0000960E EBF5                 <1>       jmp    short set_env_string_allocate_envb_retn
                              <1> 
                              <1> set_env_chk_validation4:
                              <1>       ; 11/04/2016
00009610 9D                   <1>       popf
                              <1> 
00009611 89D6                 <1>       mov    esi, edx  ; Env_Page
                              <1> 
                              <1> set_env_chk_validation5:
00009613 89DF                 <1>       mov    edi, ebx  ; ASCIIZ environment string address
00009615 0FB6CC               <1>       movzx  ecx, ah ; Variable (string) length (with '=')
                              <1> 
                              <1> set_env_chk_validation5_loop:
00009618 AC                   <1>       lodsb
00009619 AE                   <1>       scasb
0000961A 750A                 <1>       jne    short set_env_chk_validation6
0000961C E2FA                 <1>       loop   set_env_chk_validation5_loop
                              <1> 
0000961E 3C3D                 <1>       cmp    al, '='
00009620 0F8483000000         <1>         je       set_env_change_variable
                              <1> 
                              <1> set_env_chk_validation6:
00009626 08C0                 <1>       or     al, al ; 0
00009628 7403                 <1>       jz     short set_env_chk_validation7
                              <1> 
0000962A AC                   <1>       lodsb
0000962B EBF9                 <1>       jmp    short set_env_chk_validation6
                              <1> 
                              <1> set_env_chk_validation7:
0000962D 88E1                 <1>       mov    cl, ah
0000962F 01F1                 <1>       add    ecx, esi
00009631 81F9FF310900         <1>       cmp    ecx, Env_Page + Env_Page_Size - 1
                              <1>              ; 511 (4095)
                              <1>              ; strlen + '=' + 0
00009637 72DA                 <1>       jb     short set_env_chk_validation5
                              <1> 
                              <1> set_env_chk_validation8: ; variable not found
00009639 0FB6F4               <1>       movzx  esi, ah  ; variable name length (with '=')
0000963C 01DE                 <1>       add    esi, ebx ; position just after of the '='
                              <1> 
                              <1> set_env_chk_validation8_loop:
0000963E AC                   <1>       lodsb
0000963F 3C20                 <1>       cmp    al, 20h
00009641 74FB                 <1>       je     short set_env_chk_validation8_loop
00009643 72C5                 <1>       jb     short set_env_string_nothing
                              <1> 
                              <1> set_env_chk_validation9:
00009645 AC                   <1>       lodsb
00009646 3C20                 <1>       cmp    al, 20h
00009648 73FB                 <1>       jnb    short set_env_chk_validation9
                              <1> 
                              <1>       ; End of ASCIIZ environment string
                              <1> 
                              <1> set_env_add_variable:
0000964A 29DE                 <1>       sub    esi, ebx ; variable+definition length
                              <1> 
0000964C 56                   <1>       push   esi ; *****
                              <1> 
0000964D 89D6                 <1>       mov    esi, edx ; Environment page address
                              <1> 
0000964F B900020000           <1>       mov    ecx, Env_Page_Size ; 512 (4096)
                              <1> 
                              <1> set_env_add_variable_loop:
00009654 AC                   <1>       lodsb
00009655 20C0                 <1>       and    al, al
00009657 7406                 <1>       jz     short set_env_add_variable_chk1 ; 0
00009659 E2F9                 <1>       loop   set_env_add_variable_loop
                              <1> 
                              <1>       ; 11/04/2016
0000965B 884EFF               <1>       mov    [esi-1], cl ; 0
0000965E 41                   <1>       inc    ecx
                              <1> 
                              <1> set_env_add_variable_chk1:
0000965F 49                   <1>       dec    ecx
00009660 7408                 <1>       jz     short set_env_add_variable_nspc
00009662 AC                   <1>       lodsb
00009663 08C0                 <1>       or     al, al
00009665 740C                 <1>       jz     short set_env_add_variable_chk2 ; 00
00009667 49                   <1>       dec    ecx
00009668 75EA                 <1>       jnz    short set_env_add_variable_loop
                              <1> 
                              <1> set_env_add_variable_nspc: ; no space on environment page
0000966A 58                   <1>       pop    eax ; *****
0000966B B808000000           <1>       mov    eax, 8 ; No space for new environment string
00009670 F9                   <1>       stc
```

```
00009671 EB92                <1>         jmp     short set_env_string_allocate_envb_retn
                             <1> 
                             <1> set_env_add_variable_chk2:
00009673 8B0C24              <1>       mov    ecx, [esp] ; *****
00009676 4E                  <1>       dec    esi ; beginning address of the new variable
00009677 89F0                <1>       mov    eax, esi
00009679 01C8                <1>       add    eax, ecx ; string length (with CR)
0000967B 81C200020000        <1>       add    edx, Env_Page_Size ; 512 (4096)
00009681 39D0                <1>       cmp    eax, edx
00009683 77E5                <1>       ja     short set_env_add_variable_nspc
00009685 49                  <1>       dec    ecx ; except CR at the end
00009686 89CA                <1>       mov    edx, ecx ; 12/04/2016
00009688 89F7                <1>       mov    edi, esi
0000968A 893C24              <1>       mov    [esp], edi ; ***** ; Start address of new variable
0000968D 89DE                <1>       mov    esi, ebx ; ASCIIZ environment string address
0000968F F3A4                <1>       rep    movsb
00009691 28C0                <1>       sub    al, al
00009693 AA                  <1>       stosb
00009694 58                  <1>       pop    eax ; ***** ; Beginning address of new variable
00009695 81FF00320900        <1>         cmp     edi, Env_Page + Env_Page_Size ; 12/04/2016
0000969B 0F8364FFFFFF        <1>         jnb     set_env_string_allocate_envb_retn ; OK !
000096A1 880F                <1>       mov    [edi], cl ; 0
000096A3 F8                  <1>       clc    ; 13/04/2016
000096A4 E95CFFFFFF          <1>         jmp     set_env_string_allocate_envb_retn ; OK !
                             <1> 
                             <1> set_env_change_variable:
                             <1>       ; 06/04/2016
                             <1>       ; esi = Variable's address in environment page (after '=')
                             <1>       ; edi = ASCIIZ environment string address (after '=')
                             <1> 
                             <1>       ; ah = variable length from start to the '='
000096A9 8825[6C650100]      <1>       mov    [env_var_length], ah
                             <1> 
000096AF 28C9                <1>       sub    cl, cl ; ecx = 0
                             <1> 
000096B1 57                  <1>       push   edi ; *****
                             <1> 
000096B2 89F7                <1>       mov    edi, esi ; 11/04/2016
                             <1> 
                             <1> set_env_change_variable_calc1:
000096B4 AC                  <1>       lodsb
000096B5 08C0                <1>       or     al, al
000096B7 7403                <1>       jz     short set_env_change_variable_calc2
                             <1> 
000096B9 41                  <1>       inc    ecx ; length of environment string (after the '=')
                             <1> 
000096BA EBF8                <1>       jmp    short set_env_change_variable_calc1
                             <1> 
                             <1> set_env_change_variable_calc2:
000096BC 8B3424              <1>       mov    esi, [esp] ; ASCIIZ environment string address
                             <1> 
000096BF 29D2                <1>       sub    edx, edx
                             <1> 
                             <1> set_env_change_variable_calc3:
000096C1 AC                  <1>       lodsb
000096C2 3C20                <1>       cmp    al, 20h
000096C4 7203                <1>       jb     short set_env_change_variable_calc4
                             <1> 
000096C6 42                  <1>       inc    edx ; length of ASCIIZ string (after the '=')
                             <1> 
000096C7 EBF8                <1>       jmp    short set_env_change_variable_calc3
                             <1> 
                             <1> set_env_change_variable_calc4:
000096C9 C646FF00            <1>       mov    byte [esi-1], 0  ; put ZERO instead of CR
                             <1> 
000096CD 5E                  <1>       pop    esi ; ***** ; ASCIIZ string address (after '=')
                             <1> 
                             <1>       ; EDI = Old variable's address (after '=')
                             <1> 
                             <1>       ; compare the new string with the old string
000096CE 39CA                <1>       cmp    edx, ecx
000096D0 7717                <1>       ja     short set_env_change_variable_calc5 ; longer
000096D2 0F828F000000        <1>         jb      set_env_change_variable_calc9 ; shorter
                             <1> 
                             <1>       ;same length (simple copy)
000096D8 0FB6C4              <1>       movzx  eax, ah
000096DB 01C2                <1>       add    edx, eax
000096DD F7D8                <1>       neg    eax
000096DF 01F8                <1>       add    eax, edi
                             <1>       ; EAX = Start address of the variable
                             <1>       ; EDX = Variable length (without ZERO at the end of variable)
                             <1> 
000096E1 F3A4                <1>       rep    movsb
000096E3 F8                  <1>       clc    ; 13/04/2016
000096E4 E91CFFFFFF          <1>         jmp     set_env_string_allocate_envb_retn ; OK !
                             <1> 
                             <1> set_env_change_variable_calc5:
                             <1>       ; 11/04/2016
000096E9 52                  <1>       push   edx ; *****
000096EA 29CA                <1>       sub    edx, ecx ; difference ; (the new string is longer)
000096EC 89F3                <1>       mov    ebx, esi
000096EE 89FE                <1>       mov    esi, edi
                             <1> 
                             <1> set_env_change_variable_calc6:
000096F0 AC                  <1>       lodsb
000096F1 20C0                <1>       and    al, al
000096F3 75FB                <1>       jnz    short set_env_change_variable_calc6
                             <1> 
000096F5 81FE00320900        <1>       cmp    esi, Env_Page + Env_Page_Size ; 512 (4096)
000096FB 0F8369FFFFFF        <1>         jnb     set_env_add_variable_nspc
                             <1> 
00009701 89F9                <1>       mov    ecx, edi  ; current (old) variable's address
00009703 89F7                <1>       mov    edi, esi  ; next variable's address
```

```
                                   <1>
00009705 AC                        <1>       lodsb
00009706 08C0                      <1>       or     al, al
00009708 7416                      <1>       jz     short set_env_change_variable_calc8 ; 00
                                   <1>
                                   <1> set_env_change_variable_calc7:
0000970A AC                        <1>       lodsb
0000970B 20C0                      <1>       and    al, al
0000970D 75FB                      <1>       jnz    short set_env_change_variable_calc7
                                   <1>
0000970F 81FE00320900              <1>       cmp    esi, Env_Page + Env_Page_Size ; 512 (4096)
00009715 0F834FFFFFFF              <1>         jnb     set_env_add_variable_nspc
                                   <1>
0000971B AC                        <1>       lodsb
0000971C 08C0                      <1>       or     al, al
0000971E 75EA                      <1>       jnz    short set_env_change_variable_calc7
                                   <1>
                                   <1> set_env_change_variable_calc8:
00009720 4E                        <1>       dec    esi ; address of the second (last) 0 of the 00
                                   <1>
00009721 01F2                      <1>       add    edx, esi ; final position of the last 0
                                   <1>
00009723 81FA00320900              <1>       cmp    edx, Env_Page + Env_Page_Size ; 512 (4096)
00009729 0F833BFFFFFF              <1>         jnb     set_env_add_variable_nspc
                                   <1>
0000972F 89C8                      <1>       mov    eax, ecx ; old variable's address (after '=')
                                   <1>
00009731 89F1                      <1>       mov    ecx, esi
00009733 29F9                      <1>       sub    ecx, edi ; count of bytes to move forward
                                   <1>
                                   <1>       ; 13/04/2016
00009735 C60200                    <1>       mov    byte [edx], 0
00009738 89D7                      <1>       mov    edi, edx
0000973A 29F2                      <1>       sub    edx, esi ; difference (additional byte count)
0000973C 4F                        <1>       dec    edi ; the last zero address (first byte of the 00)
0000973D 89FE                      <1>       mov    esi, edi
0000973F 29D6                      <1>       sub    esi, edx ; - displacement
                                   <1>
00009741 FA                        <1>       cli    ; disable interrupts
00009742 FD                        <1>       std    ; backward
                                   <1>
00009743 F3A4                      <1>       rep    movsb ; move ECX bytes from DS:ESI to ES:EDI
                                   <1>
00009745 FC                        <1>       cld    ; forward (default)
00009746 FB                        <1>       sti    ; enable interrupts
                                   <1>
00009747 89C7                      <1>       mov    edi, eax
00009749 59                        <1>       pop    ecx ; ***** ; byte count (after '=')
0000974A 89CA                      <1>       mov    edx, ecx
0000974C 89DE                      <1>       mov    esi, ebx ; ASCIIZ string address (after '=')
0000974E 89FB                      <1>       mov    ebx, edi
                                   <1>
00009750 F3A4                      <1>       rep    movsb
                                   <1>
00009752 880F                      <1>       mov    [edi], cl ; 0 ; end of variable
                                   <1>
00009754 0FB605[6C650100]          <1>       movzx  eax, byte [env_var_length]
0000975B 01C2                      <1>       add    edx, eax ; variable length (total)
0000975D F7D8                      <1>       neg    eax
0000975F 01D8                      <1>       add    eax, ebx ; start address of the variable
00009761 F8                        <1>       clc    ; 13/04/2016
00009762 E99EFEFFFF                <1>         jmp     set_env_string_allocate_envb_retn ; OK !
                                   <1>
                                   <1> set_env_change_variable_calc9:
                                   <1>       ; 11/04/2016
00009767 21D2                      <1>       and    edx, edx ; is empty ?
00009769 753B                      <1>       jnz    short set_env_change_variable_calc15
                                   <1>
0000976B 0FB6DC                    <1>       movzx  ebx, ah
0000976E F7DB                      <1>       neg    ebx
00009770 01FB                      <1>       add    ebx, edi
                                   <1>
                                   <1>       ; EBX = Start address of the variable (in env page)
                                   <1>       ; EDX = Variable length = 0
                                   <1>
00009772 89FE                      <1>       mov    esi, edi
                                   <1>
                                   <1> set_env_change_variable_calc10:
00009774 AC                        <1>       lodsb
00009775 08C0                      <1>       or     al, al
00009777 75FB                      <1>       jnz    short set_env_change_variable_calc10
                                   <1>
00009779 B9FF310900                <1>       mov    ecx, Env_Page + Env_Page_Size - 1
                                   <1>
0000977E 39CE                      <1>       cmp    esi, ecx ; +511 (+4095)
00009780 7604                      <1>       jna    short set_env_change_variable_calc11
                                   <1>
00009782 89CE                      <1>       mov    esi, ecx
00009784 8806                      <1>       mov    [esi], al ; 0
                                   <1>
                                   <1> set_env_change_variable_calc11:
00009786 89DF                      <1>       mov    edi, ebx ; old variable's start address
                                   <1>
                                   <1> set_env_change_variable_calc12:
00009788 AC                        <1>       lodsb
00009789 AA                        <1>       stosb
0000978A 20C0                      <1>       and    al, al
0000978C 75FA                      <1>       jnz    short set_env_change_variable_calc12
0000978E 39CE                      <1>       cmp    esi, ecx
00009790 7706                      <1>       ja     short set_env_change_variable_calc13
00009792 AC                        <1>       lodsb
00009793 AA                        <1>       stosb
00009794 20C0                      <1>       and    al, al
```

```
00009796 75F0                <1>       jnz    short set_env_change_variable_calc12
                             <1>
                             <1> set_env_change_variable_calc13:
00009798 29F9                <1>       sub    ecx, edi
0000979A 7203                <1>       jb     short set_env_change_variable_calc14
0000979C 41                  <1>       inc    ecx ; 1-512 (1-4096)
0000979D F3AA                <1>       rep    stosb ; al = 0
                             <1>
                             <1> set_env_change_variable_calc14:
0000979F 29C0                <1>       sub    eax, eax ; Start address of the variable
                             <1>       ; EAX = 0 -> Variable is removed
                             <1>       ; EDX = Variable length = 0
                             <1>
000097A1 E95FFEFFFF          <1>         jmp     set_env_string_allocate_envb_retn ; OK !
                             <1>
                             <1> set_env_change_variable_calc15:
000097A6 52                  <1>       push   edx ; *****
000097A7 F7DA                <1>       neg    edx
000097A9 01CA                <1>       add    edx, ecx ; difference (the old string is longer)
000097AB 89F3                <1>       mov    ebx, esi
000097AD 89FE                <1>       mov    esi, edi
                             <1>
                             <1> set_env_change_variable_calc16:
000097AF AC                  <1>       lodsb
000097B0 20C0                <1>       and    al, al
000097B2 75FB                <1>       jnz    short set_env_change_variable_calc16
                             <1>
000097B4 B900320900          <1>       mov    ecx, Env_Page + Env_Page_Size
                             <1>
000097B9 39CE                <1>       cmp    esi, ecx ; +512 (+4096)
000097BB 7605                <1>       jna    short set_env_change_variable_calc17
                             <1>
000097BD 89CE                <1>       mov    esi, ecx
000097BF 8846FF              <1>       mov    [esi-1], al ; 0
                             <1>
                             <1> set_env_change_variable_calc17:
000097C2 89F9                <1>       mov    ecx, edi  ; current (old) variable's address
000097C4 89F7                <1>       mov    edi, esi  ; next variable's address
                             <1>
000097C6 AC                  <1>       lodsb
000097C7 08C0                <1>       or     al, al
000097C9 741D                <1>       jz     short set_env_change_variable_calc20
                             <1>
                             <1> set_env_change_variable_calc18:
000097CB AC                  <1>       lodsb
000097CC 20C0                <1>       and    al, al
000097CE 75FB                <1>       jnz    short set_env_change_variable_calc18
                             <1>
000097D0 81FE00320900        <1>       cmp    esi, Env_Page + Env_Page_Size
000097D6 720B                <1>       jb     short set_env_change_variable_calc19
000097D8 740E                <1>       je     short set_env_change_variable_calc20
                             <1>
000097DA BEFF310900          <1>       mov    esi, Env_Page + Env_Page_Size - 1
000097DF 8806                <1>       mov    [esi], al ; 0
000097E1 EB06                <1>       jmp    short set_env_change_variable_calc21
                             <1>
                             <1> set_env_change_variable_calc19:
000097E3 AC                  <1>       lodsb
000097E4 08C0                <1>       or     al, al
000097E6 75E3                <1>       jnz    short set_env_change_variable_calc18
                             <1>
                             <1> set_env_change_variable_calc20:
000097E8 4E                  <1>       dec    esi ; address of the second (last) 0 of the 00
                             <1>
                             <1> set_env_change_variable_calc21:
                             <1>       ; edx = difference (byte count)
                             <1>
000097E9 89C8                <1>       mov    eax, ecx ; old variable's address (after '=')
                             <1>
000097EB 89F1                <1>       mov    ecx, esi
000097ED 29F9                <1>       sub    ecx, edi ; count of bytes to move backward
                             <1>
000097EF 89FE                <1>       mov    esi, edi ; next variable's address
000097F1 29D7                <1>       sub    edi, edx ; (displacement)
                             <1>
000097F3 F3A4                <1>       rep    movsb
                             <1>
000097F5 880F                <1>       mov    [edi], cl ; 0 ; 00 ; end of environment variables
                             <1>
000097F7 89C7                <1>       mov    edi, eax
000097F9 5A                  <1>       pop    edx ; ***** ; byte count (after '=')
000097FA 89D1                <1>       mov    ecx, edx
000097FC 89DE                <1>       mov    esi, ebx ; ASCIIZ string address (after '=')
000097FE 89FB                <1>       mov    ebx, edi
                             <1>
00009800 F3A4                <1>       rep    movsb
                             <1>
00009802 880F                <1>       mov    [edi], cl ; 0 ; end of variable
                             <1>
00009804 0FB605[6C650100]    <1>       movzx  eax, byte [env_var_length]
0000980B 01C2                <1>       add    edx, eax ; variable length (total)
0000980D F7D8                <1>       neg    eax
0000980F 01D8                <1>       add    eax, ebx ; start address of the variable
00009811 F8                  <1>       clc    ; 13/04/2016
00009812 E9EEFDFFFF          <1>         jmp     set_env_string_allocate_envb_retn ; OK !
                             <1>
                             <1> mainprog_startup_configuration:
                             <1>       ; 22/11/2017
                             <1>       ; 06/05/2016
                             <1>       ; 14/04/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 17/09/2011 (TRDOS v1, MAINPROG.ASM)
                             <1>       ;
                             <1> loc_load_mainprog_cfg_file:
```

```
00009817 BE[190D0100]        <1>       mov   esi, MainProgCfgFile
0000981C 66B80018            <1>       mov   ax, 1800h ; Except volume label and dirs
00009820 E83EEAFFFF          <1>       call  find_first_file
00009825 7256                <1>       jc    short loc_load_mainprog_cfg_exit
                             <1> 
                             <1>       ;or   eax, eax
                             <1>       ;jz   short loc_load_mainprog_cfg_exit
                             <1> 
                             <1> loc_start_mainprog_configuration:
                             <1>       ; ESI = FindFile_DirEntry Location
                             <1>       ; EAX = File Size
                             <1> 
00009827 A3[EC580100]        <1>       mov   [MainProgCfg_FileSize], eax
                             <1> 
0000982C 668B5614            <1>       mov   dx, [esi+DirEntry_FstClusHI]
00009830 C1E210              <1>       shl   edx, 16
00009833 668B561A            <1>       mov   dx, [esi+DirEntry_FstClusLO]
00009837 8915[20650100]      <1>       mov   [csftdf_sf_cluster], edx
                             <1> 
0000983D 89C1                <1>       mov   ecx, eax
0000983F 29C0                <1>       sub   eax, eax
                             <1> 
                             <1>       ; TRDOS 386 (TRDOS v2.0)
                             <1>       ; Allocate contiguous memory block for loading the file
                             <1> 
                             <1>       ; eax = 0 (Allocate memory from the beginning)
                             <1>       ; ecx = File (Allocation) size in bytes
                             <1> 
00009841 E8DEBBFFFF          <1>       call  allocate_memory_block
00009846 7235                <1>       jc    short loc_load_mainprog_cfg_exit
                             <1> 
00009848 A3[18650100]        <1>       mov   [csftdf_sf_mem_addr], eax ; loading address
0000984D 890D[1C650100]      <1>       mov   [csftdf_sf_mem_bsize], ecx ; block size
                             <1> 
00009853 31DB                <1>       xor   ebx, ebx
                             <1>       ;mov  [csftdf_sf_rbytes], ebx ; 0, reset
                             <1> 
00009855 8A3D[FE580100]      <1>       mov   bh, [Current_Drv] ; [FindFile_Drv]
0000985B BE00010900          <1>       mov   esi, Logical_DOSDisks
00009860 01DE                <1>       add   esi, ebx
                             <1> 
00009862 8B1D[18650100]      <1>       mov   ebx, [csftdf_sf_mem_addr] ; memory block address
                             <1> 
00009868 807E0300            <1>       cmp   byte [esi+LD_FATType], 0
0000986C 7710                <1>         ja  short loc_mcfg_load_fat_file
                             <1> 
0000986E C705[28650100]0000- <1>       mov   dword [csftdf_r_size], 65536
00009876 0100                <1> 
00009878 E9A1010000          <1>         jmp     loc_mcfg_load_fs_file
                             <1> 
                             <1> loc_load_mainprog_cfg_exit:
0000987D C3                  <1>       retn
                             <1> 
                             <1> loc_mcfg_load_fat_file:
0000987E 0FB74611            <1>       movzx eax, word [esi+LD_BPB+BytesPerSec]
00009882 0FB64E13            <1>       movzx ecx, byte [esi+LD_BPB+SecPerClust]
00009886 F7E1                <1>       mul   ecx
00009888 A3[28650100]        <1>       mov   [csftdf_r_size], eax
                             <1> 
                             <1> loc_mcfg_load_fat_file_next:
0000988D E822010000          <1>       call  mcfg_read_fat_file_sectors
00009892 0F8206010000        <1>         jc      mcfg_deallocate_mem
                             <1> 
00009898 09D2                <1>       or    edx, edx ; edx > 0 -> EOF
0000989A 74F1                <1>       jz    short loc_mcfg_load_fat_file_next
                             <1> 
                             <1> loc_mcfg_load_fat_file_ok:
                             <1>       ; 06/05/2016
0000989C C705[BC650100]-     <1>       mov   dword [mainprog_return_addr], loc_mcfg_ci_return_addr
000098A2 [5F990000]          <1> 
                             <1>       ;
000098A6 8B35[18650100]      <1>       mov   esi, [csftdf_sf_mem_addr]
000098AC 8935[F0580100]      <1>       mov   [MainProgCfg_LineOffset], esi
                             <1> 
000098B2 A1[EC580100]        <1>       mov   eax, [MainProgCfg_FileSize]
000098B7 89C2                <1>       mov   edx, eax
000098B9 01F2                <1>       add   edx, esi
                             <1> 
                             <1> loc_mcfg_process_next_line_check:
000098BB 89C1                <1>       mov   ecx, eax
                             <1> 
000098BD 803E2A              <1>       cmp   byte [esi], "*" ; Remark sign
000098C0 7503                <1>       jne   short loc_mcfg_process_next_line
000098C2 46                  <1>       inc   esi
000098C3 EB17                <1>       jmp   short loc_move_mainprog_cfg_nl1
                             <1> 
                             <1> loc_mcfg_process_next_line:
000098C5 83F94F              <1>       cmp   ecx, 79
000098C8 7605                <1>       jna   short loc_start_mainprog_cfg_process
                             <1> 
000098CA B94F000000          <1>       mov   ecx, 79
                             <1> 
                             <1> loc_start_mainprog_cfg_process:
000098CF BF[AE590100]        <1>       mov   edi, CommandBuffer
                             <1> 
                             <1> loc_move_mainprog_cfg_line:
000098D4 AC                  <1>       lodsb
000098D5 3C20                <1>       cmp   al, 20h
000098D7 720C                <1>       jb    short loc_move_mainprog_cfg_nl2
000098D9 AA                  <1>       stosb
000098DA E2F8                <1>       loop  loc_move_mainprog_cfg_line
                             <1> 
                             <1> loc_move_mainprog_cfg_nl1:
```

```
000098DC 39D6                <1>       cmp    esi, edx ; + configuration file size
000098DE 7312                <1>       jnb    short loc_end_of_mainprog_cfg_line
000098E0 AC                  <1>       lodsb
000098E1 3C20                <1>       cmp    al, 20h
000098E3 73F7                <1>       jnb    short loc_move_mainprog_cfg_nl1
                             <1>
                             <1> loc_move_mainprog_cfg_nl2:
000098E5 39D6                <1>       cmp    esi, edx
000098E7 7309                <1>       jnb    short loc_end_of_mainprog_cfg_line
000098E9 8A06                <1>       mov    al, [esi]
000098EB 3C20                <1>       cmp    al, 20h
000098ED 7703                <1>       ja     short loc_end_of_mainprog_cfg_line
000098EF 46                  <1>       inc    esi
000098F0 EBF3                <1>       jmp    short loc_move_mainprog_cfg_nl2
                             <1>
                             <1> loc_end_of_mainprog_cfg_line:
000098F2 C60700              <1>       mov    byte [edi], 0
                             <1>
000098F5 8935[F0580100]      <1>       mov    [MainProgCfg_LineOffset], esi
                             <1>
                             <1>       ; 22/11/2017
000098FB BE[B6590100]        <1>       mov    esi, CommandBuffer + 8
00009900 29FE                <1>       sub    esi, edi
00009902 7606                <1>       jna    short loc_move_mainprog_cfg_command
00009904 30C0                <1>       xor    al, al
                             <1> loc_mainprog_cfg_clear_chrs:
00009906 AA                  <1>       stosb
00009907 4E                  <1>       dec    esi
00009908 75FC                <1>       jnz    short loc_mainprog_cfg_clear_chrs
                             <1>
                             <1> loc_move_mainprog_cfg_command:
0000990A BE[AE590100]        <1>       mov    esi, CommandBuffer
0000990F 89F7                <1>       mov    edi, esi
00009911 31DB                <1>       xor    ebx, ebx
                             <1>       ;xor   ecx, ecx
00009913 30C9                <1>       xor    cl, cl
                             <1>
                             <1> loc_move_mcfg_first_cmd_char:
00009915 8A041E              <1>       mov    al, [esi+ebx]
00009918 FEC3                <1>       inc    bl
0000991A 3C20                <1>       cmp    al, 20h
0000991C 7712                <1>       ja     short loc_move_mcfg_cmd_capitalizing
0000991E 7237                <1>       jb     short loc_move_mcfg_cmd_arguments_ok
00009920 80FB4F              <1>       cmp    bl, 79
00009923 72F0                <1>       jb     short loc_move_mcfg_first_cmd_char
00009925 EB30                <1>       jmp    short loc_move_mcfg_cmd_arguments_ok
                             <1>
                             <1> loc_move_mcfg_next_cmd_char:
00009927 8A041E              <1>       mov    al, [esi+ebx]
0000992A FEC3                <1>       inc    bl
0000992C 3C20                <1>       cmp    al, 20h
0000992E 7614                <1>       jna    short loc_move_mcfg_cmd_ok
                             <1>
                             <1> loc_move_mcfg_cmd_capitalizing:
00009930 3C61                <1>       cmp    al, 61h ; 'a'
00009932 7206                <1>       jb     short loc_move_mcfg_cmd_caps_ok
00009934 3C7A                <1>       cmp    al, 7Ah ; 'z'
00009936 7702                <1>       ja     short loc_move_mcfg_cmd_caps_ok
00009938 24DF                <1>       and    al, 0DFh ; sub      al, 'a'-'A'
                             <1>
                             <1> loc_move_mcfg_cmd_caps_ok:
0000993A AA                  <1>       stosb
0000993B FEC1                <1>       inc    cl
0000993D 80FB4F              <1>       cmp    bl, 79
00009940 72E5                <1>       jb     short loc_move_mcfg_next_cmd_char
00009942 EB13                <1>       jmp    short loc_move_mcfg_cmd_arguments_ok
                             <1>
                             <1> loc_move_mcfg_cmd_ok:
00009944 30C0                <1>       xor    al, al ; 0
                             <1>
                             <1> loc_move_mcfg_cmd_arguments:
00009946 8807                <1>       mov    [edi], al
00009948 47                  <1>       inc    edi
00009949 80FB4F              <1>       cmp    bl, 79
0000994C 7309                <1>       jnb    short loc_move_mcfg_cmd_arguments_ok
0000994E 8A041E              <1>       mov    al, [esi+ebx]
00009951 FEC3                <1>       inc    bl
00009953 3C20                <1>       cmp    al, 20h
00009955 73EF                <1>       jnb    short loc_move_mcfg_cmd_arguments
                             <1>
                             <1> loc_move_mcfg_cmd_arguments_ok:
00009957 C60700              <1>       mov    byte [edi], 0
                             <1>
                             <1> loc_mcfg_process_cmd_interpreter:
0000995A E825E0FFFF          <1>       call    command_interpreter
                             <1>
                             <1> loc_mcfg_ci_return_addr:
0000995F A1[EC580100]        <1>       mov    eax, [MainProgCfg_FileSize]
00009964 89C2                <1>       mov    edx, eax
00009966 8B35[F0580100]      <1>       mov    esi, [MainProgCfg_LineOffset]
0000996C 01F2                <1>       add    edx, esi
0000996E 0305[18650100]      <1>       add    eax, [csftdf_sf_mem_addr]
00009974 29F0                <1>       sub    eax, esi
00009976 0F873FFFFFFF        <1>         ja      loc_mcfg_process_next_line_check
                             <1>
0000997C E81D000000          <1>       call   mcfg_deallocate_mem
                             <1>
00009981 B94F000000          <1>       mov    ecx, 79 ; 80 ?
00009986 BF[AE590100]        <1>       mov    edi, CommandBuffer
0000998B 30C0                <1>       xor    al, al
0000998D F3AA                <1>       rep    stosb
                             <1>
                             <1>       ; 06/05/2016
```

```
0000998F BE[6F190100]        <1>       mov    esi, nextline
00009994 E8C4C9FFFF          <1>       call   print_msg
00009999 E963D6FFFF          <1>       jmp    dos_prompt
                             <1> 
                             <1> mcfg_deallocate_mem:
0000999E A1[18650100]        <1>       mov    eax, [csftdf_sf_mem_addr] ; start address
000099A3 8B0D[1C650100]      <1>       mov    ecx, [csftdf_sf_mem_bsize] ; block size
                             <1>       ;call  deallocate_memory_block
                             <1>       ;retn
000099A9 E983BCFFFF          <1>       jmp    deallocate_memory_block
                             <1> 
                             <1> mcfg_read_file_sectors:
                             <1>       ; 14/04/2016
000099AE 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
000099B2 7669                <1>         jna  short mcfg_read_fs_file_sectors
                             <1> 
                             <1> mcfg_read_fat_file_sectors:
                             <1>       ; return:
                             <1>       ;   CF = 0 & EDX > 0 -> END OF FILE
                             <1>       ;   CF = 0 & EDX = 0 -> not EOF
                             <1>       ;   CF = 1 -> read error (error code in AL)
                             <1> 
                             <1> mcfg_read_fat_file_secs_0:
000099B4 8B15[EC580100]      <1>       mov    edx, [MainProgCfg_FileSize]
000099BA 2B15[30650100]      <1>       sub    edx, [csftdf_sf_rbytes]
000099C0 3B15[28650100]      <1>       cmp    edx, [csftdf_r_size]
000099C6 7306                <1>       jnb    short mcfg_read_fat_file_secs_1
000099C8 8915[28650100]      <1>       mov    [csftdf_r_size], edx
                             <1> 
                             <1> mcfg_read_fat_file_secs_1:
000099CE A1[28650100]        <1>       mov    eax, [csftdf_r_size]
000099D3 29D2                <1>       sub    edx, edx
000099D5 0FB74E11            <1>       movzx  ecx, word [esi+LD_BPB+BytesPerSec]
000099D9 01C8                <1>       add    eax, ecx
000099DB 48                  <1>       dec    eax
000099DC F7F1                <1>       div    ecx
000099DE 89C1                <1>       mov    ecx, eax ; sector count
000099E0 A1[20650100]        <1>       mov    eax, [csftdf_sf_cluster]
                             <1> 
                             <1>       ; EBX = memory block address (current)
                             <1> 
000099E5 E88C230000          <1>       call   read_fat_file_sectors
000099EA 7230                <1>       jc     short mcfg_read_fat_file_secs_3
                             <1> 
                             <1>       ; EBX = next memory address
                             <1> 
000099EC A1[30650100]        <1>       mov    eax, [csftdf_sf_rbytes]
000099F1 0305[28650100]      <1>       add    eax, [csftdf_r_size]
000099F7 8B15[EC580100]      <1>       mov    edx, [MainProgCfg_FileSize]
000099FD 39D0                <1>       cmp    eax, edx
000099FF 731B                <1>       jnb    short mcfg_read_fat_file_secs_3 ; edx > 0
00009A01 A3[30650100]        <1>       mov    [csftdf_sf_rbytes], eax
                             <1> 
00009A06 53                  <1>       push   ebx ; *
                             <1>       ; get next cluster (csftdf_r_size! bytes)
00009A07 A1[20650100]        <1>       mov    eax, [csftdf_sf_cluster]
00009A0C E837210000          <1>       call   get_next_cluster
00009A11 5B                  <1>       pop    ebx ; *
00009A12 7301                <1>       jnc    short mcfg_read_fat_file_secs_2
                             <1> 
                             <1>       ;mov   eax, 17; Read error !
00009A14 C3                  <1>       retn
                             <1> 
                             <1> mcfg_read_fat_file_secs_2:
00009A15 29D2                <1>       sub    edx, edx ; 0
00009A17 A3[20650100]        <1>       mov    [csftdf_sf_cluster], eax ; next cluster
                             <1> 
                             <1> mcfg_read_fat_file_secs_3:
00009A1C C3                  <1>       retn
                             <1> 
                             <1> mcfg_read_fs_file_sectors:
00009A1D C3                  <1>       retn
                             <1> 
                             <1> loc_mcfg_load_fs_file:
00009A1E C3                  <1>       retn
                             <1> 
                             <1> load_and_execute_file:
                             <1>       ; 04/01/2017
                             <1>       ; 06/05/2016, 07/05/2016, 11/05/2016
                             <1>       ; 23/04/2016, 24/04/2016
                             <1>       ; 22/04/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 05/11/2011
                             <1>       ; (TRDOS v1, CMDINTR.ASM, 'cmp_cmd_run', 'cmp_cmd_external')
                             <1>       ; ('loc_run_check_filename')
                             <1>       ; 29/08/2011
                             <1>       ; 10/09/2011
                             <1>       ; INPUT->
                             <1>       ;      ESI = Path Name address (CommandBuffer address)
                             <1>       ; OUTPUT ->
                             <1>       ;      none (error message will be shown if an error will occur)
                             <1>       ;
                             <1>       ; (EAX, EBX, ECX, EDX, ESI, EDI, EBP will be changed)
                             <1>       ;
                             <1> loc_run_check_filename:
00009A1F 803E20              <1>       cmp    byte [esi], 20h
00009A22 0F822BE3FFFF        <1>       jb     loc_cmd_failed
00009A28 7703                <1>       ja     short loc_run_check_filename_ok
00009A2A 46                  <1>       inc    esi
00009A2B EBF2                <1>       jmp    short loc_run_check_filename
                             <1> 
                             <1> loc_run_check_filename_ok:
00009A2D C605[5F590100]00    <1>       mov    byte [CmdArgStart], 0 ; reset
00009A34 56                  <1>       push   esi ; *
```

```
                             <1> loc_run_get_first_arg_pos:
00009A35 46                  <1>       inc    esi
00009A36 8A06                <1>       mov    al, [esi]
00009A38 3C20                <1>       cmp    al, 20h
00009A3A 77F9                <1>       ja     short loc_run_get_first_arg_pos
00009A3C C60600              <1>       mov    byte [esi], 0
                             <1> loc_run_get_external_arg_pos:
                             <1>       ; 11/05/2016
00009A3F 46                  <1>       inc    esi
00009A40 8A06                <1>       mov    al, [esi]
00009A42 3C20                <1>       cmp    al, 20h
00009A44 760C                <1>       jna    short loc_run_parse_path_name
00009A46 89F0                <1>       mov    eax, esi
00009A48 2D[AE590100]        <1>       sub    eax, CommandBuffer
00009A4D A2[5F590100]        <1>       mov    byte [CmdArgStart], al
                             <1> loc_run_parse_path_name:
00009A52 5E                  <1>       pop    esi ; *
00009A53 BF[A2620100]        <1>       mov    edi, FindFile_Drv
00009A58 E8D7090000          <1>       call   parse_path_name
00009A5D 0F82F0E2FFFF        <1>       jc     loc_cmd_failed
                             <1>
                             <1> loc_run_check_filename_exists:
00009A63 BE[E4620100]        <1>       mov    esi, FindFile_Name
00009A68 803E20              <1>       cmp    byte [esi], 20h
00009A6B 0F86E2E2FFFF        <1>       jna    loc_cmd_failed
                             <1>
                             <1> loc_run_check_exe_filename_ext:
00009A71 E890020000          <1>       call   check_prg_filename_ext
00009A76 0F82D7E2FFFF        <1>       jc     loc_cmd_failed
                             <1>
                             <1> loc_run_check_exe_filename_ext_ok:
00009A7C 66A3[BA650100]      <1>       mov    word [EXE_ID], ax
                             <1>
                             <1> loc_run_drv:
00009A82 C605[B9650100]00    <1>       mov    byte [Run_Manual_Path], 0
00009A89 A1[F8580100]        <1>       mov    eax, [Current_Dir_FCluster]
00009A8E A3[B4650100]        <1>         mov     [Run_CDirFC], eax
                             <1>       ;
00009A93 8A35[FE580100]      <1>       mov    dh, [Current_Drv]
00009A99 8835[5E610100]      <1>       mov    [RUN_CDRV], dh
                             <1>
00009A9F 8A15[A2620100]      <1>       mov    dl, [FindFile_Drv]
00009AA5 38F2                <1>       cmp    dl, dh
00009AA7 7412                <1>       je     short loc_run_change_directory
                             <1>
00009AA9 8005[B9650100]02    <1>       add    byte [Run_Manual_Path], 2
                             <1>
00009AB0 E80BD4FFFF          <1>       call   change_current_drive
00009AB5 0F82C3E2FFFF        <1>       jc     loc_run_cmd_failed
                             <1>
                             <1> loc_run_change_directory:
00009ABB 803D[A3620100]20    <1>       cmp    byte [FindFile_Directory], 20h
00009AC2 7623                <1>       jna    short loc_run_find_executable_file
                             <1>
00009AC4 FE05[B9650100]      <1>       inc    byte [Run_Manual_Path]
                             <1>
00009ACA FE05[D30C0100]      <1>       inc    byte [Restore_CDIR]
                             <1>
00009AD0 BE[A3620100]        <1>       mov    esi, FindFile_Directory
00009AD5 30E4                <1>       xor    ah, ah ; CD_COMMAND sign -> 0
00009AD7 E842030000          <1>       call   change_current_directory
00009ADC 0F829CE2FFFF        <1>       jc     loc_run_cmd_failed
                             <1>
                             <1> loc_run_change_prompt_dir_string:
00009AE2 E857020000          <1>       call   change_prompt_dir_string
                             <1>
                             <1> loc_run_find_executable_file:
00009AE7 66C705[B8650100]00- <1>       mov    word [Run_Auto_Path], 0
00009AEF 00                  <1>
                             <1>
                             <1> loc_run_find_executable_file_next:
00009AF0 BE[E4620100]        <1>       mov    esi, FindFile_Name
                             <1> loc_run_find_program_file_next:
00009AF5 66B80018            <1>       mov    ax, 1800h ; Except volume label and dirs
00009AF9 E865E7FFFF          <1>       call   find_first_file
                             <1>       ; ESI = Directory Entry (FindFile_DirEntry) Location
                             <1>       ; EDI = Directory Buffer Directory Entry Location
                             <1>       ; EAX = File size
00009AFE 0F835C010000        <1>       jnc    loc_load_and_run_file
                             <1>
00009B04 3C02                <1>       cmp    al, 2 ; file not found
00009B06 0F8572E2FFFF        <1>       jne    loc_run_cmd_failed
                             <1>
00009B0C 66A1[BA650100]      <1>       mov    ax, word [EXE_ID]
00009B12 80FC2E              <1>       cmp    ah, '.' ; File name has extension sign
00009B15 7424                <1>       je     short loc_run_check_auto_path
                             <1>
00009B17 08C0                <1>       or     al, al
00009B19 7520                <1>       jnz    short loc_run_check_auto_path
                             <1>
00009B1B 80FC08              <1>       cmp    ah, 8 ; count of file name chars
00009B1E 771B                <1>       ja     short loc_run_check_auto_path
                             <1>
                             <1> loc_run_change_file_ext_to_prg:
00009B20 0FB6DC              <1>       movzx  ebx, ah ; count of file name chars
00009B23 BE[E4620100]        <1>       mov    esi, FindFile_Name
00009B28 01F3                <1>       add    ebx, esi
                             <1>       ; 07/05/2016
00009B2A C7032E505247        <1>       mov    dword [ebx],  '.PRG'
00009B30 66C705[BA650100]50- <1>       mov    word [EXE_ID], 'P.'
00009B38 2E                  <1>
00009B39 EBBA                <1>       jmp    short loc_run_find_program_file_next
                             <1>
```

```
                            <1> loc_run_check_auto_path:
                            <1>       ; NOTE: /// 07/05/2016 ///
                            <1>       ; If the path is given, value of byte [Run_Manual_Path]
                            <1>       ; will not be ZERO. If so, file searching by using
                            <1>       ; Automatic Path (via 'PATH' environment variable)
                            <1>       ; will not be applicable, because the program file
                            <1>       ; is already/absolutely not found.
                            <1>
00009B3B A0[B9650100]       <1>       mov    al, [Run_Manual_Path]
00009B40 08C0               <1>       or     al, al
00009B42 0F850BE2FFFF       <1>       jnz    loc_cmd_failed
                            <1>
                            <1> loc_run_check_auto_path_again:
00009B48 66833D[B8650100]FF <1>       cmp    word [Run_Auto_Path], 0FFFFh
                            <1>              ; 0FFFFh = Not a valid run path (in ENV block)
00009B50 0F83FDE1FFFF       <1>       jnb    loc_cmd_failed
                            <1>       ; xor  al, al
00009B56 BE[9F0D0100]       <1>       mov    esi, Cmd_Path ; 'PATH'
00009B5B BF[FE590100]       <1>       mov    edi, TextBuffer
00009B60 E848F9FFFF         <1>       call   get_environment_string
00009B65 730E               <1>       jnc    short loc_run_chk_filename_ext_again
00009B67 66C705[B8650100]FF- <1>      mov    word [Run_Auto_Path], 0FFFFh ; invalid
00009B6F FF                 <1>
00009B70 E9DEE1FFFF         <1>       jmp    loc_cmd_failed
                            <1>
                            <1> loc_run_chk_filename_ext_again:
00009B75 89C1               <1>       mov    ecx, eax ; string length (with zero tail)
00009B77 49                 <1>       dec    ecx ; without zero tail
00009B78 66A1[BA650100]     <1>       mov    ax, [EXE_ID]
00009B7E 80FC2E             <1>       cmp    ah, '.'
00009B81 740E               <1>       je     short loc_run_chk_auto_path_pos
                            <1>
                            <1> loc_run_change_file_ext_to_noext_again:
00009B83 0FB6DC             <1>       movzx  ebx, ah
00009B86 BE[E4620100]       <1>       mov    esi, FindFile_Name
00009B8B 01F3               <1>       add    ebx, esi
00009B8D 29C0               <1>       sub    eax, eax
00009B8F 8903               <1>       mov    [ebx], eax ; 0 ; erase extension (.PRG)
                            <1>
                            <1> loc_run_chk_auto_path_pos:
                            <1>       ;movzx eax,  word [Run_Auto_Path]
00009B91 66A1[B8650100]     <1>       mov    ax, [Run_Auto_Path]
00009B97 39C8               <1>       cmp    eax, ecx ; ecx = string length (except zero tail)
00009B99 0F83B4E1FFFF       <1>       jnb    loc_cmd_failed
                            <1>       ;or    eax, eax
00009B9F 6609C0             <1>       or     ax, ax
00009BA2 7502               <1>       jnz    short loc_run_auto_path_pos_move
00009BA4 B005               <1>       mov    al, 5
                            <1>
                            <1> loc_run_auto_path_pos_move:
00009BA6 89FE               <1>       mov    esi, edi ; offset TextBuffer
00009BA8 01C6               <1>       add    esi, eax
                            <1>
                            <1> loc_run_auto_path_pos_space_loop:
00009BAA AC                 <1>       lodsb
00009BAB 3C20               <1>       cmp    al, 20h
00009BAD 74FB               <1>       je     short loc_run_auto_path_pos_space_loop
00009BAF 0F829EE1FFFF       <1>       jb     loc_cmd_failed
00009BB5 AA                 <1>       stosb
                            <1> loc_run_auto_path_pos_move_next:
00009BB6 AC                 <1>       lodsb
00009BB7 3C3B               <1>       cmp    al, ';'
00009BB9 7414               <1>       je     short loc_run_auto_path_pos_move_last_byte
00009BBB 3C20               <1>       cmp    al, 20h
00009BBD 74F7               <1>       je     short loc_run_auto_path_pos_move_next
00009BBF 7203               <1>       jb     short loc_byte_ptr_end_of_path
00009BC1 AA                 <1>       stosb
00009BC2 EBF2               <1>       jmp    short loc_run_auto_path_pos_move_next
                            <1>
                            <1> loc_byte_ptr_end_of_path:
00009BC4 66C705[B8650100]FF- <1>      mov    word [Run_Auto_Path], 0FFFFh ; end of path
00009BCC FF                 <1>
00009BCD EB0D               <1>       jmp    short loc_run_auto_path_move_ok
                            <1>
                            <1> loc_run_auto_path_pos_move_last_byte:
00009BCF 89F0               <1>       mov    eax, esi
00009BD1 2D[FE590100]       <1>       sub    eax, TextBuffer
00009BD6 66A3[B8650100]     <1>       mov    [Run_Auto_Path], ax ; next path position
                            <1>
                            <1> loc_run_auto_path_move_ok:
00009BDC 4F                 <1>       dec    edi
00009BDD B02F               <1>       mov    al, '/'
00009BDF 3807               <1>       cmp    [edi], al
00009BE1 7403               <1>       je     short loc_run_auto_path_move_file_name
00009BE3 47                 <1>       inc    edi
00009BE4 8807               <1>       mov    [edi], al
                            <1>
                            <1> loc_run_auto_path_move_file_name:
00009BE6 47                 <1>       inc    edi
00009BE7 BE[E4620100]       <1>       mov    esi, FindFile_Name
                            <1>
                            <1> loc_run_auto_path_move_fn_loop:
00009BEC AC                 <1>       lodsb
00009BED AA                 <1>       stosb
00009BEE 08C0               <1>       or     al, al
00009BF0 75FA               <1>       jnz    short loc_run_auto_path_move_fn_loop
                            <1>
00009BF2 BE[FE590100]       <1>       mov    esi, TextBuffer
00009BF7 BF[A2620100]       <1>       mov    edi, FindFile_Drv
00009BFC E833080000         <1>       call   parse_path_name
00009C01 0F824CE1FFFF       <1>       jc     loc_cmd_failed
                            <1>
00009C07 8A35[FE580100]     <1>       mov    dh, [Current_Drv]
```

```
00009C0D 8A15[A2620100]      <1>       mov   dl, [FindFile_Drv]
00009C13 38F2                <1>       cmp   dl, dh
00009C15 740B                <1>       je    short loc_run_change_directory_again
                             <1>
00009C17 E8A4D2FFFF          <1>       call  change_current_drive
00009C1C 0F825CE1FFFF        <1>       jc    loc_run_cmd_failed
                             <1>
                             <1> loc_run_change_directory_again:
00009C22 803D[A3620100]20    <1>       cmp   byte [FindFile_Directory], 20h
00009C29 761D                <1>       jna   short loc_load_executable_cdir_chk_again
                             <1>
00009C2B FE05[D30C0100]      <1>       inc   byte [Restore_CDIR]
00009C31 BE[A3620100]        <1>       mov   esi, FindFile_Directory
00009C36 30E4                <1>       xor   ah, ah ; CD_COMMAND sign -> 0
00009C38 E8E1010000          <1>       call  change_current_directory
00009C3D 0F823BE1FFFF        <1>       jc    loc_run_cmd_failed
                             <1>
                             <1> loc_run_chg_prompt_dir_str_again:
00009C43 E8F6000000          <1>       call  change_prompt_dir_string
                             <1>
                             <1> loc_load_executable_cdir_chk_again:
00009C48 A1[F8580100]        <1>       mov   eax, [Current_Dir_FCluster]
00009C4D 3B05[B4650100]      <1>       cmp   eax, [Run_CDirFC]
00009C53 0F8597FEFFFF        <1>       jne   loc_run_find_executable_file_next
00009C59 30C0                <1>       xor   al, al ; 0
00009C5B E9E8FEFFFF          <1>       jmp   loc_run_check_auto_path_again
                             <1>
                             <1> loc_load_and_run_file:
                             <1>       ; 13/11/2017
                             <1>       ; 04/01/2017
                             <1>       ; 23/04/2016
00009C60 BE[E4620100]        <1>       mov   esi, FindFile_Name
00009C65 BF[FE590100]        <1>       mov   edi, TextBuffer
                             <1>
                             <1>       ; 24/04/2016
00009C6A 31D2                <1>       xor   edx, edx
00009C6C 668915[4A040300]    <1>       mov   word [argc], dx ; 0
00009C73 8915[8C030300]      <1>       mov   dword [u.nread], edx ; 0
                             <1>
                             <1> loc_load_and_run_file_1:
00009C79 AC                  <1>       lodsb
00009C7A AA                  <1>       stosb
00009C7B FF05[8C030300]      <1>       inc   dword [u.nread]
00009C81 20C0                <1>       and   al, al
00009C83 75F4                <1>       jnz   short loc_load_and_run_file_1
                             <1>
00009C85 A0[5F590100]        <1>       mov   al, [CmdArgStart]
00009C8A 20C0                <1>       and   al, al
00009C8C 7445                <1>       jz    short loc_load_and_run_file_7
                             <1>
00009C8E 0FB6F0              <1>       movzx esi, al ; 11/05/2016
00009C91 B950000000          <1>       mov   ecx, 80
00009C96 29F1                <1>       sub   ecx, esi
00009C98 81C6[AE590100]      <1>       add   esi, CommandBuffer
                             <1>
00009C9E 66FF05[4A040300]    <1>       inc   word [argc] ; 11/05/2016
                             <1>
                             <1> loc_load_and_run_file_2:
00009CA5 AC                  <1>       lodsb
00009CA6 3C20                <1>       cmp   al, 20h
00009CA8 7717                <1>       ja    short loc_load_and_run_file_5
00009CAA 721E                <1>       jb    short loc_load_and_run_file_6
                             <1>
                             <1> loc_load_and_run_file_3:
00009CAC 803E20              <1>       cmp   byte [esi], 20h
00009CAF 7707                <1>       ja    short loc_load_and_run_file_4
00009CB1 7217                <1>       jb    short loc_load_and_run_file_6
00009CB3 46                  <1>       inc   esi
00009CB4 E2F6                <1>       loop  loc_load_and_run_file_3
00009CB6 EB12                <1>       jmp   short loc_load_and_run_file_6
                             <1>
                             <1> loc_load_and_run_file_4:
00009CB8 28C0                <1>       sub   al, al ; 0
00009CBA 66FF05[4A040300]    <1>       inc   word [argc]
                             <1> loc_load_and_run_file_5:
00009CC1 AA                  <1>       stosb
00009CC2 FF05[8C030300]      <1>       inc   dword [u.nread]
00009CC8 E2DB                <1>       loop  loc_load_and_run_file_2
                             <1>
                             <1> loc_load_and_run_file_6:
00009CCA 30C0                <1>       xor   al, al ; 0
00009CCC AA                  <1>       stosb
00009CCD FF05[8C030300]      <1>       inc   dword [u.nread]
                             <1> loc_load_and_run_file_7:
00009CD3 8807                <1>       mov   [edi], al ; 0
00009CD5 66FF05[4A040300]    <1>       inc   word [argc] ; 24/04/2016
00009CDC FF05[8C030300]      <1>       inc   dword [u.nread] ; 24/04/2016
00009CE2 BE[FE590100]        <1>       mov   esi, TextBuffer
00009CE7 8B15[10630100]      <1>       mov   edx, [FindFile_DirEntry+DirEntry_FileSize]
00009CED 66A1[08630100]      <1>       mov   ax, [FindFile_DirEntry+DirEntry_FstClusHI]
00009CF3 C1E010              <1>       shl   eax, 16 ; 13/11/2017
00009CF6 66A1[0E630100]      <1>       mov   ax, [FindFile_DirEntry+DirEntry_FstClusLO]
                             <1>       ; EAX = First Cluster number
                             <1>       ; EDX = File Size
                             <1>       ; ESI = Argument list address
                             <1>       ; [argc] = argument count
                             <1>       ; [u.nread] = argument list length
00009CFC E89D420000          <1>       call  load_and_run_file ; trdosk6.s
                             <1>         ;jc loc_run_cmd_failed ; 04/01/2017
                             <1> loc_load_and_run_file_8: ; 06/05/2016
00009D01 E98BE9FFFF          <1>       jmp   loc_file_rw_restore_retn
                             <1>
                             <1> check_prg_filename_ext:
                             <1>      ; 23/04/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>      ; 10/09/2011
                             <1>      ; (TRDOS v1, CMDINTR.ASM, 'proc_check_exe_filename_ext')
                             <1>      ; 14/11/2009
                             <1>      ; INPUT ->
                             <1>      ;     ESI = Dot File Name
                             <1>      ; OUTPUT ->
                             <1>      ;     cf = 0 -> EXE_ID in AL
                             <1>      ;     ESI = Last char + 1 position
                             <1>      ;     cf = 1 -> Invalid executable file name
                             <1>      ;     or no file name extension if AH<=8
                             <1>      ;     AL = Last file name char
                             <1>      ;     cf = 0 -> AL='P' (PRG), AL=0 (no extension)
                             <1>      ;
                             <1>      ; (Modified registers: EAX, ESI)
                             <1>
00009D06 30E4                 <1>      xor    ah, ah
                             <1> loc_run_check_filename_ext:
00009D08 AC                   <1>      lodsb
00009D09 3C21                 <1>      cmp    al, 21h
00009D0B 7229                 <1>      jb     short loc_check_exe_fn_retn
00009D0D FEC4                 <1>      inc    ah
00009D0F 3C2E                 <1>      cmp    al, '.'
00009D11 75F5                 <1>      jne    short loc_run_check_filename_ext
                             <1>
                             <1> loc_run_check_filename_ext_dot:
00009D13 80FC02               <1>      cmp    ah, 2 ; .??? is not valid
00009D16 88C4                 <1>      mov    ah, al ; '.'
00009D18 7219                 <1>      jb     short loc_check_prg_fn_retn
                             <1>
                             <1> loc_run_check_filename_ext_dot_ok:
00009D1A AC                   <1>      lodsb
00009D1B 24DF                 <1>      and    al, 0DFh
                             <1>
                             <1> loc_run_check_filename_ext_prg:
00009D1D 3C50                 <1>      cmp    al, 'P'
00009D1F 7212                 <1>      jb     short loc_check_prg_fn_retn
00009D21 7711                 <1>      ja     short loc_check_prg_fn_stc
00009D23 AC                   <1>      lodsb
00009D24 24DF                 <1>      and    al, 0DFh
00009D26 3C52                 <1>      cmp    al, 'R'
00009D28 750A                 <1>      jne    short loc_check_prg_fn_stc
00009D2A AC                   <1>      lodsb
00009D2B 24DF                 <1>      and    al, 0DFh
00009D2D 3C47                 <1>      cmp    al, 'G'
00009D2F 7503                 <1>      jne    short loc_check_prg_fn_stc
                             <1>
00009D31 B050                 <1>      mov    al, 'P'
                             <1> loc_check_prg_fn_retn:
00009D33 C3                   <1>      retn
                             <1>
                             <1> loc_check_prg_fn_stc:
00009D34 F9                   <1>      stc
00009D35 C3                   <1>      retn
                             <1>
                             <1> loc_check_exe_fn_retn:
00009D36 28C0                 <1>      sub    al, al ; 0
00009D38 C3                   <1>      retn
                             <1>
                             <1> find_and_list_files:
00009D39 C3                   <1>      retn
                             <1> set_exec_arguments:
00009D3A C3                   <1>      retn
                             <1> delete_fs_directory:
00009D3B 31C0                 <1>      xor eax, eax
00009D3D C3                   <1>      retn
                                 %include 'trdosk4.s' ; 24/01/2016
                             <1> ; ***************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - Directory Functions : trdosk4.s
                             <1> ; ---------------------------------------------------------------------------
                             <1> ; Last Update: 29/12/2017
                             <1> ; ---------------------------------------------------------------------------
                             <1> ; Beginning: 24/01/2016
                             <1> ; ---------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ---------------------------------------------------------------------------
                             <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
                             <1> ; DIR.ASM (09/10/2011)
                             <1> ; ***************************************************************************
                             <1>
                             <1> ; DIR.ASM  [ TRDOS KERNEL - COMMAND EXECUTER SECTION - DIRECTORY FUNCTIONS ]
                             <1> ; (c) 2004-2010  Erdogan TAN  [ 17/01/2004 ]  Last Update: 09/10/2011
                             <1> ; FILE.ASM [ FILE FUNCTIONS ] Last Update: 09/10/2011
                             <1>
                             <1> change_prompt_dir_string:
                             <1>      ; 05/10/2016
                             <1>      ; 24/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>      ; 27/03/2011
                             <1>      ; 09/10/2009
                             <1>      ; INPUT/OUTPUT => none
                             <1>      ; this procedure changes current directory string/text
                             <1>      ; 2005
                             <1>
00009D3E BE[5F610100]         <1>      mov    esi, PATH_Array
                             <1> change_prompt_dir_str: ; 05/10/2016 (call from 'set_working_path')
00009D43 BF[02590100]         <1>      mov    edi, Current_Directory
00009D48 8A25[FC580100]       <1>      mov    ah, [Current_Dir_Level]
00009D4E E807000000           <1>      call   set_current_directory_string
00009D53 880D[5D590100]       <1>      mov    [Current_Dir_StrLen], cl
                             <1>
00009D59 C3                   <1>      retn
                             <1>
                             <1> set_current_directory_string:
```

```
                             <1>       ; 24/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 27/03/2011
                             <1>       ; 09/10/2009
                             <1>       ; INPUT:
                             <1>       ;    ESI = Path Array Address
                             <1>       ;    EDI = Current Directory String Buffer
                             <1>       ;    AH = Current Directory Level
                             <1>       ; OUTPUT => EAX, EBX, ESI will be changed
                             <1>       ;    EDI will be same with input
                             <1>       ;    ECX = Current Directory String Length
                             <1>
00009D5A 57                  <1>       push    edi
00009D5B 80FC00              <1>       cmp     ah, 0
00009D5E 7652                <1>       jna    short pass_write_path
00009D60 83C610              <1>       add    esi, 16
00009D63 89F3                <1>       mov    ebx, esi
                             <1> loc_write_path:
00009D65 B908000000          <1>       mov    ecx, 8
                             <1> path_write_dirname1:
00009D6A AC                  <1>       lodsb
00009D6B 3C20                <1>       cmp    al, 20h
00009D6D 7612                <1>       jna    short pass_write_dirname1
00009D6F AA                  <1>       stosb
00009D70 81FF[5C590100]      <1>       cmp    edi, End_Of_Current_Dir_Str
00009D76 733A                <1>       jnb    short pass_write_path
00009D78 E2F0                <1>       loop   path_write_dirname1
00009D7A 803E20              <1>       cmp    byte [esi], 20h
00009D7D 7624                <1>       jna    short pass_write_dirname2
00009D7F EB0A                <1>       jmp     short loc_put_dot_cont_ext
                             <1> pass_write_dirname1:
00009D81 89DE                <1>       mov    esi, ebx
00009D83 83C608              <1>       add    esi, 8
00009D86 803E20              <1>       cmp    byte [esi], 20h
00009D89 7618                <1>       jna    short pass_write_dirname2
                             <1> loc_put_dot_cont_ext:
00009D8B C6072E              <1>       mov    byte [edi], "."
                             <1>       ;mov   ecx, 3
00009D8E B103                <1>       mov    cl, 3
                             <1> loc_check_dir_name_ext:
00009D90 AC                  <1>       lodsb
00009D91 47                  <1>       inc    edi
00009D92 3C20                <1>       cmp    al, 20h
00009D94 760D                <1>       jna    short pass_write_dirname2
00009D96 8807                <1>       mov    [edi], al
00009D98 81FF[5C590100]      <1>       cmp    edi, End_Of_Current_Dir_Str
00009D9E 7312                <1>       jnb    short pass_write_path
00009DA0 E2EE                <1>       loop    loc_check_dir_name_ext
00009DA2 47                  <1>       inc    edi
                             <1> pass_write_dirname2:
00009DA3 FECC                <1>       dec    ah
00009DA5 740B                <1>       jz      short pass_write_path
00009DA7 83C310              <1>       add    ebx, 16
00009DAA 89DE                <1>       mov    esi, ebx
00009DAC C6072F              <1>       mov    byte [edi],"/"
00009DAF 47                  <1>       inc    edi
00009DB0 EBB3                <1>       jmp    short loc_write_path
                             <1> pass_write_path:
00009DB2 C60700              <1>       mov    byte [edi], 0
00009DB5 47                  <1>       inc    edi
00009DB6 89F9                <1>       mov    ecx, edi
00009DB8 5F                  <1>       pop    edi
00009DB9 29F9                <1>       sub    ecx, edi
                             <1>       ; ECX = Current Directory String Length
00009DBB C3                  <1>       retn
                             <1>
                             <1> get_current_directory:
                             <1>       ; 15/10/2016
                             <1>       ; 14/02/2016
                             <1>       ; 24/01/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 27/03/2011
                             <1>       ;
                             <1>       ; INPUT-> ESI = Current Directory Buffer
                             <1>       ;         DL = TRDOS Logical Dos Drive Number + 1
                             <1>       ;              (0= Default/Current Drive)
                             <1>       ;
                             <1>       ;   Note: Required dir buffer length may be <= 92 bytes
                             <1>       ;         for TRDOS (7*12 name chars + 7 slash + 0)
                             <1>       ; OUTPUT ->  ESI = Current Directory Buffer
                             <1>       ;            EAX, EBX, ECX, EDX, EDI will be changed
                             <1>       ;            CX/CL = Current Directory String Length
                             <1>       ;          DL = Drive Number (0 based)
                             <1>       ;            (If input is 0, output is current drv number)
                             <1>       ;            DH = same with input
                             <1>       ;   cf = 0 -> AL = 0
                             <1>       ;   cf = 1 -> error code in AL
                             <1>
                             <1> loc_get_current_drive_0:
00009DBC 80FA00              <1>       cmp    dl, 0
00009DBF 7708                <1>       ja     short loc_get_current_drive_1
00009DC1 8A15[FE580100]      <1>       mov    dl, [Current_Drv]
00009DC7 EB17                <1>       jmp    short loc_get_current_drive_2
                             <1> loc_get_current_drive_1:
00009DC9 FECA                <1>       dec    dl
00009DCB 3A15[D20C0100]      <1>       cmp    dl, [Last_DOS_DiskNo]
00009DD1 760D                <1>       jna    short loc_get_current_drive_2
00009DD3 B80F000000          <1>       mov    eax, 0Fh ; Invalid drive (Drive not ready!)
00009DD8 F5                  <1>       cmc    ; stc
00009DD9 C3                  <1>       retn
                             <1>
                             <1> loc_get_current_drive_not_ready_retn:
00009DDA 5E                  <1>       pop    esi
                             <1>       ;mov   eax, 15
00009DDB 66B80F00            <1>       mov    ax, 15 ; Drive not ready
```

```
00009DDF C3                  <1>       retn
                             <1>
                             <1> loc_get_current_drive_2:
00009DE0 31C0                <1>       xor    eax, eax
00009DE2 88D4                <1>       mov    ah, dl
00009DE4 56                  <1>       push   esi
00009DE5 BE00010900          <1>       mov    esi, Logical_DOSDisks
00009DEA 01C6                <1>       add    esi, eax
00009DEC 8A06                <1>       mov    al, [esi+LD_Name]
00009DEE 3C41                <1>       cmp    al, 'A'
00009DF0 72E8                <1>       jb     short loc_get_current_drive_not_ready_retn
                             <1>
00009DF2 8A667F              <1>       mov    ah, [esi+LD_CDirLevel]
00009DF5 08E4                <1>       or     ah, ah
00009DF7 7506                <1>       jnz    short loc_get_current_drive_3
                             <1>
                             <1>       ;xor   ah, ah ; mov ah, 0
00009DF9 8826                <1>       mov    [esi], ah
00009DFB 31C9                <1>       xor    ecx, ecx
00009DFD EB1C                <1>       jmp    short loc_get_current_drive_4
                             <1>
                             <1> loc_get_current_drive_3:
00009DFF BF[5F610100]        <1>         mov     edi, PATH_Array
00009E04 57                  <1>       push   edi
00009E05 81C680000000        <1>       add    esi, LD_CurrentDirectory
00009E0B B920000000          <1>       mov    ecx, 32
00009E10 F3A5                <1>       rep    movsd
00009E12 5E                  <1>       pop    esi ; Path Array Address
00009E13 5F                  <1>       pop    edi ; pushed esi (current dir buffer offset)
                             <1>       ;
00009E14 E841FFFFFF          <1>       call   set_current_directory_string
00009E19 89FE                <1>       mov    esi, edi
                             <1>
                             <1> loc_get_current_drive_4:
00009E1B 30C0                <1>       xor    al, al
00009E1D C3                  <1>       retn
                             <1>
                             <1> change_current_directory:
                             <1>       ; 19/02/2016
                             <1>       ; 11/02/2016
                             <1>       ; 10/02/2016
                             <1>       ; 08/02/2016
                             <1>       ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 18/09/2011 (DIR.ASM, 09/10/2011)
                             <1>       ; 04/10/2009
                             <1>       ; 2005
                             <1>       ; INPUT ->
                             <1>       ;      ESI = Directory string
                             <1>       ;      ah = CD command (CDh = save current dir string)
                             <1>       ; OUTPUT ->
                             <1>       ;      EDI = DOS Drive Description Table
                             <1>       ;      cf = 1 -> error
                             <1>       ;         EAX = Error code
                             <1>       ;      cf = 0 -> successful
                             <1>       ;         ESI = PATH_Array
                             <1>       ;         EAX = Current Directory First Cluster
                             <1>       ;
                             <1>       ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
                             <1>
00009E1E 8825[ED610100]      <1>       mov    [CD_COMMAND], ah
00009E24 803E2F              <1>       cmp    byte [esi], '/'
00009E27 7505                <1>       jne    short loc_ccd_cdir_level
00009E29 46                  <1>       inc    esi
00009E2A 30C0                <1>       xor    al, al
00009E2C EB05                <1>       jmp    short loc_ccd_parse_path_name
                             <1> loc_ccd_cdir_level:
00009E2E A0[FC580100]        <1>       mov    al, [Current_Dir_Level]
                             <1> loc_ccd_parse_path_name:
00009E33 88C4                <1>       mov    ah, al
00009E35 BF[5F610100]        <1>       mov    edi, PATH_Array
                             <1>
                             <1> ; Reset directory levels > cdir level
                             <1>       ; is this required !?
                             <1>       ;
                             <1>       ; Relations:
                             <1>       ; MAINPROG.ASM (pass_ccdrv_reset_cdir_FAT_fcluster)
                             <1>       ; proc_parse_dir_name,
                             <1>       ; proc_change_current_directory (this procedure)
                             <1>       ; proc_change_prompt_dir_string
                             <1>
00009E3A 0FB6C8              <1>       movzx  ecx, al
00009E3D FEC1                <1>       inc    cl
00009E3F C0E104              <1>       shl    cl, 4
00009E42 01CF                <1>       add    edi, ecx
00009E44 B107                <1>       mov    cl, 7
00009E46 28C1                <1>       sub    cl, al
00009E48 C0E102              <1>       shl    cl, 2
00009E4B 89C3                <1>       mov    ebx, eax
00009E4D 31C0                <1>       xor    eax, eax ; 0
00009E4F F3AB                <1>       rep    stosd
00009E51 89D8                <1>       mov    eax, ebx
                             <1>
00009E53 BF[5F610100]        <1>       mov    edi, PATH_Array
                             <1>
00009E58 803E20              <1>       cmp    byte [esi], 20h
00009E5B F5                  <1>       cmc
00009E5C 7305                <1>       jnc    short pass_ccd_parse_dir_name
                             <1>
                             <1>              ; ESI = Path name
                             <1>              ; AL = CCD_Level
00009E5E E872010000          <1>         call    parse_dir_name
                             <1>              ; AL = CCD_Level
                             <1>              ; AH = Last_Dir_Level
```

```
                                <1>              ; (EDI = PATH_Array)
                                <1>
                                <1> pass_ccd_parse_dir_name:
00009E63 9C                     <1>       pushf
                                <1>
                                <1>       ;mov   [CCD_Level], al
                                <1>         ;mov [Last_Dir_Level], ah
00009E64 66A3[E3610100]         <1>       mov    [CCD_Level], ax
                                <1>
00009E6A 31DB                   <1>       xor    ebx, ebx
00009E6C 8A3D[FE580100]         <1>       mov    bh, [Current_Drv]
00009E72 BE00010900             <1>       mov    esi, Logical_DOSDisks
00009E77 01DE                   <1>       add    esi, ebx
                                <1>
00009E79 9D                     <1>       popf
00009E7A 720A                   <1>       jc     short loc_ccd_bad_path_name_retn
                                <1>
00009E7C 8935[DF610100]         <1>       mov    [CCD_DriveDT], esi
                                <1>
00009E82 3C07                   <1>       cmp    al, 7
00009E84 7209                   <1>       jb     short loc_ccd_load_child_dir
                                <1>
                                <1> loc_ccd_bad_path_name_retn:
00009E86 87F7                   <1>       xchg   esi, edi
00009E88 B813000000             <1>       mov    eax, 19 ; Bad directory/path name
00009E8D F9                     <1>       stc
                                <1> loc_ccd_retn_p:
00009E8E C3                     <1>       retn
                                <1>
                                <1> loc_ccd_load_child_dir:
                                <1>       ; AL = CCD_Level
00009E8F 08C0                   <1>       or     al, al
00009E91 7468                   <1>       jz     short loc_ccd_load_root_dir
                                <1>
00009E93 6689C1                 <1>       mov    cx, ax
00009E96 C0E004                 <1>       shl    al, 4
00009E99 0FB6F0                 <1>       movzx  esi, al
00009E9C 01FE                   <1>       add    esi, edi  ; offset PATH_Array
                                <1>
00009E9E 8B460C                 <1>       mov    eax, [esi+12]
00009EA1 38E9                   <1>       cmp    cl, ch
00009EA3 0F84FA000000           <1>         je       loc_ccd_load_sub_directory
00009EA9 A3[F8580100]           <1>       mov    [Current_Dir_FCluster], eax
                                <1>
                                <1> loc_ccd_load_child_dir_next:
00009EAE 83C610                 <1>       add    esi, 16 ; DOS DirEntry Format FileName Address
                                <1>
                                <1>       ; Directory attribute : 10h
00009EB1 B010                   <1>       mov    al, 00010000b ; 10h (Attrib AND mask)
                                <1>       ;mov   ah, 11001000b ; C8h
                                <1>       ; Volume name attribute: 8h
00009EB3 B408                   <1>       mov    ah, 00001000b ; 08h (Attrib NAND, AND --> zero mask)
                                <1>
00009EB5 6631C9                 <1>       xor    cx, cx
00009EB8 E8B5010000             <1>       call   locate_current_dir_file
00009EBD 7353                   <1>       jnc    short loc_ccd_set_dir_cluster_ptr
                                <1>
                                <1>        ; 19/02/2016
                                <1>       ;mov   edi, [CCD_DriveDT]
00009EBF 8A25[E3610100]         <1>       mov    ah, [CCD_Level]
00009EC5 803D[ED610100]CD       <1>       cmp    byte [CD_COMMAND], 0CDh ;'CD' command or another
00009ECC 7509                   <1>       jne    short loc_ccd_load_child_dir_err
                                <1>       ; It is better to save recent successful part
                                <1>       ; of the (requested) path as current directory.
                                <1>       ; (Otherwise the path would be reset to back
                                <1>       ; on the next 'CD' command.)
00009ECE 88E1                   <1>       mov    cl, ah
00009ED0 50                     <1>       push   eax
00009ED1 E8E3000000             <1>       call   loc_ccd_save_current_dir
00009ED6 58                     <1>       pop    eax
                                <1> loc_ccd_load_child_dir_err:
00009ED7 3C03                   <1>       cmp    al, 3  ; AL = 2 => File not found error
00009ED9 7202                   <1>       jb     short loc_ccd_path_not_found_retn
00009EDB F9                     <1>       stc
00009EDC C3                     <1>       retn
                                <1>
                                <1> loc_ccd_path_not_found_retn:
00009EDD B003                   <1>       mov    al, 3  ; Path not found
00009EDF C3                     <1>       retn
                                <1>
                                <1> loc_ccd_load_FAT_root_dir:
00009EE0 803D[FD580100]02       <1>       cmp    byte [Current_FATType], 2
00009EE7 776B                   <1>       ja     short loc_ccd_load_FAT32_root_dir
                                <1>
                                <1>       ;mov   esi, [CCD_DriveDT]
                                <1>       ;push  esi
00009EE9 E8B51D0000             <1>       call   load_FAT_root_directory
                                <1>       ;pop   edi ; Dos Drv Description Table
                                <1>
00009EEE 89F7                   <1>       mov    edi, esi
00009EF0 BE[5F610100]           <1>       mov    esi, PATH_Array
00009EF5 7297                   <1>       jc     short loc_ccd_retn_p
                                <1>
00009EF7 31C0                   <1>       xor    eax, eax
00009EF9 EB78                   <1>         jmp  short loc_ccd_set_cdfc
                                <1>
                                <1> loc_ccd_load_root_dir:
00009EFB 803D[FD580100]01       <1>       cmp    byte [Current_FATType], 1
00009F02 73DC                   <1>       jnb    short loc_ccd_load_FAT_root_dir
                                <1>
                                <1> loc_ccd_load_FS_root_dir:
00009F04 E8611E0000             <1>       call   load_FS_root_directory
00009F09 EB5C                   <1>       jmp    short pass_ccd_load_FAT_sub_directory
```

```
                             <1> 
                             <1> loc_ccd_load_FS_sub_directory_next:
00009F0B E85B1E0000          <1>       call  load_FS_sub_directory
00009F10 EB1F                <1>       jmp   short pass_ccd_set_dir_cluster_ptr
                             <1> 
                             <1> loc_ccd_set_dir_cluster_ptr:
                             <1>       ; EDI = Directory Entry
00009F12 668B4714            <1>       mov   ax, [edi+20] ; First Cluster High Word
00009F16 C1E010              <1>       shl   eax, 16
00009F19 668B471A            <1>       mov   ax, [edi+26] ; First Cluster Low Word
                             <1> 
00009F1D 8B35[DF610100]      <1>       mov   esi, [CCD_DriveDT]
00009F23 803D[FD580100]01    <1>       cmp   byte [Current_FATType], 1
00009F2A 72DF                <1>       jb    short loc_ccd_load_FS_sub_directory_next
                             <1>       ;push esi
00009F2C E8FD1D0000          <1>       call  load_FAT_sub_directory
                             <1>       ;pop  edi ; Dos Drv Description Table
                             <1> 
                             <1> pass_ccd_set_dir_cluster_ptr:
                             <1>       ;mov  edi, esi
00009F31 BE[5F610100]        <1>       mov   esi, PATH_Array
00009F36 7264                <1>       jc    short loc_ccd_retn_c
                             <1> 
00009F38 A1[2D610100]        <1>       mov   eax, [DirBuff_Cluster]
                             <1> 
00009F3D FE05[E3610100]      <1>       inc   byte [CCD_Level]
00009F43 0FB61D[E3610100]    <1>       movzx ebx, byte [CCD_Level]
00009F4A C0E304              <1>       shl   bl, 4 ; * 16 (<= 128)
00009F4D 01DE                <1>       add   esi, ebx ; 19/02/2016
00009F4F 89460C              <1>       mov   [esi+12], eax
00009F52 EB1F                <1>       jmp   short loc_ccd_set_cdfc
                             <1> 
                             <1> loc_ccd_load_FAT32_root_dir:
00009F54 BE[5F610100]        <1>       mov   esi, PATH_Array
00009F59 8B460C              <1>       mov   eax, [esi+12]
00009F5C 8B35[DF610100]      <1>       mov   esi, [CCD_DriveDT]
                             <1> 
                             <1> loc_ccd_load_FAT_sub_directory:
                             <1>       ;push esi
00009F62 E8C71D0000          <1>       call  load_FAT_sub_directory
                             <1>       ;pop  edi ; Dos Drv Description Table
                             <1> 
                             <1> pass_ccd_load_FAT_sub_directory:
                             <1>       ;mov  edi, esi
00009F67 BE[5F610100]        <1>       mov   esi, PATH_Array
00009F6C 722E                <1>       jc    short loc_ccd_retn_c
                             <1> 
00009F6E A1[2D610100]        <1>       mov   eax, [DirBuff_Cluster]
                             <1> 
                             <1> loc_ccd_set_cdfc:
00009F73 8A0D[E3610100]      <1>       mov   cl, [CCD_Level]
00009F79 880D[FC580100]      <1>       mov   [Current_Dir_Level], cl
00009F7F A3[F8580100]        <1>       mov   [Current_Dir_FCluster], eax
                             <1> 
00009F84 8A2D[E4610100]      <1>       mov   ch, [Last_Dir_Level]
00009F8A 38E9                <1>       cmp   cl, ch
00009F8C 0F821CFFFFFF        <1>       jb    loc_ccd_load_child_dir_next
                             <1> 
00009F92 803D[ED610100]CD    <1>       cmp   byte [CD_COMMAND], 0CDh ;'CD' command or another
00009F99 741E                <1>       je    short loc_ccd_save_current_dir
                             <1> 
                             <1>         ; jne -> don't save, restore (the previous cdir) later !
                             <1>         ; (saving the cdir would prevent previous cdir restoration!)
                             <1> 
00009F9B F8                  <1>       clc
                             <1> 
                             <1> loc_ccd_retn_c:
00009F9C 8B3D[DF610100]      <1>       mov   edi, [CCD_DriveDT]
00009FA2 C3                  <1>       retn
                             <1> 
                             <1> loc_ccd_load_sub_directory:
00009FA3 8B35[DF610100]      <1>       mov   esi, [CCD_DriveDT]
00009FA9 803D[FD580100]01    <1>       cmp   byte [Current_FATType], 1
00009FB0 73B0                <1>       jnb   short loc_ccd_load_FAT_sub_directory
00009FB2 E8B41D0000          <1>       call  load_FS_sub_directory
00009FB7 EBAE                <1>       jmp   short pass_ccd_load_FAT_sub_directory
                             <1> 
                             <1> loc_ccd_save_current_dir:
00009FB9 BE[5F610100]        <1>       mov   esi, PATH_Array ; 19/02/2016
00009FBE 8B3D[DF610100]      <1>       mov   edi, [CCD_DriveDT]
00009FC4 57                  <1>       push  edi
00009FC5 83C77F              <1>         add     edi, LD_CDirLevel
00009FC8 880F                <1>       mov   [edi], cl
00009FCA 47                  <1>       inc   edi ; LD_CurrentDirectory
00009FCB 56                  <1>       push  esi
                             <1>       ;mov  ecx, 32  ; always < 65536 (in this procedure)
00009FCC 66B92000            <1>       mov   cx, 32
00009FD0 F3A5                <1>       rep   movsd
                             <1>       ; Current directory has been saved to
                             <1>       ; the DOS drive description table, cdir area !
00009FD2 5E                  <1>       pop   esi  ; PATH_Array
00009FD3 5F                  <1>       pop   edi  ; Dos Drv Description Table
                             <1> 
00009FD4 C3                  <1>       retn
                             <1> 
                             <1> parse_dir_name:
                             <1>       ; 11/02/2016
                             <1>       ; 10/02/2016
                             <1>       ; 07/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 18/09/2011
                             <1>       ; 17/10/2009
                             <1>       ; INPUT ->
                             <1>       ;      ESI = ASCIIZ Directory String Address
```

```
                              <1>        ;      AL = Current Directory Level
                              <1>        ;      EDI = Destination Adress
                              <1>        ;           (8 levels, each one 12+4 byte)
                              <1>        ; OUTPUT ->
                              <1>        ;      EDI = Dir Entry Formatted Array
                              <1>        ;           with zero cluster pointer at the last level
                              <1>        ;      AH = Last Dir Level
                              <1>        ;      AL = Current Dir Level
                              <1>        ;
                              <1>        ; (esi, ebx, ecx will be changed)
                              <1>
                              <1>        ;mov   [PATH_Array_Ptr], edi
00009FD5 88C4                 <1>        mov    ah, al
00009FD7 66A3[84620100]       <1>        mov    [PATH_CDLevel], ax
                              <1> repeat_ppdn_check_slash:
00009FDD AC                   <1>        lodsb
00009FDE 3C2F                 <1>        cmp    al, '/'
00009FE0 74FB                 <1>        je     short repeat_ppdn_check_slash
00009FE2 3C21                 <1>        cmp    al, 21h
00009FE4 7219                 <1>        jb     short loc_ppdn_retn
00009FE6 57                   <1>        push   edi
                              <1> loc_ppdn_get_dir_name:
00009FE7 B90C000000           <1>        mov    ecx, 12
00009FEC BF[86620100]         <1>        mov    edi, Dir_File_Name
                              <1> repeat_ppdn_get_dir_name:
00009FF1 AA                   <1>        stosb
00009FF2 AC                   <1>        lodsb
00009FF3 3C2F                 <1>        cmp    al, '/'
00009FF5 740A                 <1>        je     short loc_check_level_dot_conv_dir_name
00009FF7 3C20                 <1>        cmp    al, 20h
00009FF9 7605                 <1>        jna    short loc_ppdn_end_of_path_scan
00009FFB E2F4                 <1>        loop   repeat_ppdn_get_dir_name
00009FFD 5F                   <1>        pop    edi
00009FFE F9                   <1>        stc
                              <1> loc_ppdn_retn:
00009FFF C3                   <1>        retn
                              <1>
                              <1> loc_ppdn_end_of_path_scan:
0000A000 4E                   <1>        dec    esi
                              <1> loc_check_level_dot_conv_dir_name:
0000A001 31C0                 <1>        xor    eax, eax
0000A003 AA                   <1>        stosb
0000A004 89F3                 <1>        mov    ebx, esi
0000A006 BE[86620100]         <1>        mov    esi, Dir_File_Name
0000A00B AC                   <1>        lodsb
                              <1> repeat_ppdn_name_check_dot:
0000A00C 3C2E                 <1>        cmp    al, '.'
0000A00E 7509                 <1>        jne    short loc_ppdn_convert_sub_dir_name
                              <1> repeat_ppdn_name_dot_dot:
0000A010 AC                   <1>        lodsb
0000A011 3C2E                 <1>        cmp    al, '.'
0000A013 743E                 <1>        je     short loc_ppdn_dot_dot
0000A015 3C21                 <1>        cmp    al, 21h
0000A017 7226                 <1>        jb     short pass_ppdn_convert_sub_dir_name
                              <1> loc_ppdn_convert_sub_dir_name:
0000A019 8A25[85620100]       <1>        mov    ah, [PATH_Level]
0000A01F 80FC07               <1>        cmp    ah, 7
0000A022 731B                 <1>        jnb    short pass_ppdn_convert_sub_dir_name
0000A024 FEC4                 <1>        inc    ah
0000A026 8825[85620100]       <1>        mov    [PATH_Level], ah
0000A02C BE[86620100]         <1>        mov    esi, Dir_File_Name
                              <1>        ;mov   edi, [PATH_Array_Ptr]
0000A031 B010                 <1>        mov    al, 16
0000A033 F6E4                 <1>        mul    ah
0000A035 8B3C24               <1>        mov    edi, [esp]
                              <1>        ;push  edi
0000A038 01C7                 <1>        add    edi, eax
0000A03A E82A030000           <1>        call   convert_file_name
                              <1>        ;pop   edi
                              <1> pass_ppdn_convert_sub_dir_name:
0000A03F 89DE                 <1>        mov    esi, ebx
                              <1> repeat_ppdn_check_last_slash:
0000A041 AC                   <1>        lodsb
0000A042 3C2F                 <1>        cmp    al, '/'
0000A044 74FB                 <1>        je     short repeat_ppdn_check_last_slash
0000A046 3C21                 <1>        cmp    al, 21h
0000A048 739D                 <1>        jnb    short loc_ppdn_get_dir_name
                              <1> end_of_parse_dir_name:
0000A04A 5F                   <1>        pop    edi
0000A04B F5                   <1>        cmc
                              <1>        ;mov   al, [PATH_CDLevel]
                              <1>        ;mov   ah, [PATH_Level]
0000A04C 66A1[84620100]       <1>        mov    ax, [PATH_CDLevel]
0000A052 C3                   <1>        retn
                              <1>
                              <1> loc_ppdn_dot_dot:
0000A053 AC                   <1>        lodsb
0000A054 3C21                 <1>        cmp    al, 21h
0000A056 73F2                 <1>        jnb    short end_of_parse_dir_name
                              <1> loc_ppdn_dot_dot_prev_level:
0000A058 66A1[84620100]       <1>        mov    ax, [PATH_CDLevel]
0000A05E 80EC01               <1>        sub    ah, 1
0000A061 80D400               <1>        adc    ah, 0
0000A064 38E0                 <1>        cmp    al, ah
0000A066 7602                 <1>        jna    short pass_ppdn_set_al_to_ah
0000A068 88E0                 <1>        mov    al, ah
                              <1> pass_ppdn_set_al_to_ah:
0000A06A 66A3[84620100]       <1>        mov    [PATH_CDLevel], ax
0000A070 EBCD                 <1>        jmp    short pass_ppdn_convert_sub_dir_name
                              <1>
                              <1> locate_current_dir_file:
                              <1>        ; 20/11/2017
                              <1>        ; 14/02/2016
```

```
                             <1>       ; 13/02/2016
                             <1>       ; 10/02/2016
                             <1>       ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 14/08/2010
                             <1>       ; 19/09/2009
                             <1>         ; 2005
                             <1>       ; INPUT ->
                             <1>       ;      ESI = DOS DirEntry Format FileName Address
                             <1>       ;      AL = Attributes Mask
                             <1>       ;      (<AL AND EntryAttrib> must be equal to AL)
                             <1>       ;      AH = Negative Attributes Mask (If AH>0)
                             <1>       ;      (<AH AND EntryAttrib> must be ZERO)
                             <1>       ;      CH > 0 Find First Free Dir Entry or Deleted Entry
                             <1>       ;      CL = 0 -> Return the First Free Dir Entry
                             <1>       ;      CL = E5h -> Return the 1st deleted entry
                             <1>       ;      CL = FFh -> Return the 1st deleted or free entry
                             <1>       ;      CL > 0 and CL <> E5h and CL <> FFh -> Return the first
                             <1>       ;           proper entry (which fits with Atributes Masks)
                             <1>       ;      CX = 0 Find Valid File/Directory/VolumeName
                             <1>       ;      ? = Any One Char
                             <1>       ;      * = Every Chars
                             <1>       ; OUTPUT ->
                             <1>       ;      EDI = Directory Entry Address (in Directory Buffer)
                             <1>       ;      ESI = DOS DirEntry Format FileName Address
                             <1>       ;      CF = 0 -> No Error, Proper Entry,
                             <1>       ;      DL = Attributes
                             <1>       ;      DH = Previous Entry Attr (LongName Check)
                             <1>       ;      AL > 0 -> Ambiguous filename wildcard "?" used
                             <1>       ;      AH > 0 -> Ambiguous filename wildcard "*" used
                             <1>       ;      AX = 0 -> Filename full fits with directory entry
                             <1>       ;      CH = The 1st Name Char of Current Dir Entry
                             <1>       ;      CF = 1 -> Proper entry not found, Error Code in EAX/AL
                             <1>       ;      CL = 0 and CH = 0 -> Free Entry (End Of Dir)
                             <1>       ;      CL = 0 and CH = E5h -> Deleted Entry fits with filters
                             <1>       ;      CL > 0 -> Entry not found, CH invalid
                             <1>       ;      CF = 0 ->
                             <1>       ;      EBX = Current Directory Entry Index/Number (BX)
                             <1>
                             <1>       ;mov   word [DirBuff_EntryCounter], 0 ; Zero Based
                             <1>
0000A072 8935[E7610100]      <1>       mov    [CDLF_FNAddress], esi
0000A078 66A3[E5610100]      <1>       mov    [CDLF_AttributesMask], ax
0000A07E 66890D[EB610100]    <1>       mov    [CDLF_DEType], cx
                             <1>
0000A085 31DB                <1>       xor    ebx, ebx
0000A087 881D[FC610100]      <1>       mov    [PreviousAttr], bl ; 0  ; 13/02/2016
                             <1>
0000A08D 8A3D[FE580100]      <1>       mov    bh, [Current_Drv]
0000A093 381D[28610100]      <1>       cmp    byte [DirBuff_ValidData], bl ; 0
0000A099 761D                <1>       jna    short loc_lcdf_reload_current_dir2
0000A09B 8A1D[26610100]      <1>         mov     bl, [DirBuff_DRV]
0000A0A1 80EB41              <1>       sub    bl, 'A'
0000A0A4 38DF                <1>       cmp    bh, bl
0000A0A6 750E                <1>       jne    short loc_lcdf_reload_current_dir1
0000A0A8 8B15[2D610100]      <1>       mov    edx, [DirBuff_Cluster]
0000A0AE 3B15[F8580100]      <1>       cmp    edx, [Current_Dir_FCluster]
0000A0B4 7412                <1>       je     short loc_cdir_locatefile_search
                             <1>
                             <1> loc_lcdf_reload_current_dir1:
0000A0B6 30DB                <1>       xor    bl, bl
                             <1> loc_lcdf_reload_current_dir2:
0000A0B8 89DE                <1>       mov    esi, ebx
0000A0BA 81C600010900        <1>         add     esi, Logical_DOSDisks
0000A0C0 E874000000          <1>       call   reload_current_directory
0000A0C5 735D                <1>       jnc    short loc_locatefile_search_again
0000A0C7 C3                  <1>       retn
                             <1>
                             <1> loc_cdir_locatefile_search:
0000A0C8 31DB                <1>       xor    ebx, ebx
0000A0CA 55                  <1>       push   ebp ; 20/11/2017
0000A0CB E8A6000000          <1>       call   find_directory_entry
0000A0D0 5D                  <1>       pop    ebp ; 20/11/2017
0000A0D1 7349                <1>       jnc    short loc_cdir_locate_file_retn
                             <1>
                             <1> loc_locatefile_check_stc_reason:
0000A0D3 08ED                <1>       or     ch, ch
0000A0D5 7444                <1>       jz     short loc_cdir_locate_file_stc_retn
                             <1>
                             <1> loc_locatefile_check_next_entryblock:
0000A0D7 8A3D[FE580100]      <1>       mov    bh, [Current_Drv]
0000A0DD 28DB                <1>       sub    bl, bl
0000A0DF 0FB7F3              <1>       movzx  esi, bx
0000A0E2 81C600010900        <1>         add     esi, Logical_DOSDisks
                             <1>
0000A0E8 803D[FC580100]00    <1>       cmp    byte [Current_Dir_Level], 0
0000A0EF 760A                <1>       jna    short loc_locatefile_check_FAT_type
                             <1>
0000A0F1 803D[FD580100]01    <1>       cmp    byte [Current_FATType], 1
0000A0F8 730A                <1>       jnb    short loc_locatefile_load_subdir_cluster
0000A0FA C3                  <1>       retn
                             <1>
                             <1> loc_locatefile_check_FAT_type:
0000A0FB 803D[FD580100]03    <1>       cmp    byte [Current_FATType], 3
0000A102 7218                <1>       jb     short loc_cdir_locate_file_retn
                             <1>
                             <1> loc_locatefile_load_subdir_cluster:
0000A104 A1[2D610100]        <1>       mov    eax, [DirBuff_Cluster]
0000A109 E83A1A0000          <1>       call   get_next_cluster
0000A10E 730D                <1>       jnc    short loc_locatefile_next_cluster
0000A110 09C0                <1>       or     eax, eax
0000A112 7507                <1>       jnz    short loc_locatefile_drive_not_ready_read_err
0000A114 F9                  <1>       stc
                             <1> loc_locatefile_file_notfound:
```

```
0000A115 B802000000          <1>       mov    eax, 2 ; File/Directory/VolName not found
0000A11A C3                  <1>       retn
                             <1>
                             <1> loc_locatefile_drive_not_ready_read_err:
                             <1>       ;mov   eax, 17 ;Drive not ready or read error
                             <1> loc_cdir_locate_file_stc_retn:
0000A11B F5                  <1>       cmc ;stc
                             <1> loc_cdir_locate_file_retn:
0000A11C C3                  <1>       retn
                             <1>
                             <1> loc_locatefile_next_cluster:
0000A11D E80C1C0000          <1>       call   load_FAT_sub_directory
                             <1>       ;jc    short loc_locatefile_drive_not_ready_read_err
0000A122 72F8                <1>       jc     short loc_cdir_locate_file_retn
                             <1>
                             <1> loc_locatefile_search_again:
0000A124 8B35[E7610100]      <1>       mov    esi, [CDLF_FNAddress]
0000A12A 66A1[E5610100]      <1>       mov    ax, [CDLF_AttributesMask]
0000A130 668B0D[EB610100]    <1>       mov    cx, [CDLF_DEType]
0000A137 EB8F                <1>       jmp    short loc_cdir_locatefile_search
                             <1>
                             <1> reload_current_directory:
                             <1>       ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 13/06/2010
                             <1>       ; 22/09/2009
                             <1>         ;
                             <1>       ; INPUT ->
                             <1>       ;      ESI = Dos drive description table address
                             <1>
                             <1>       ;mov   al, [esi+LD_FATType]
0000A139 A0[FD580100]        <1>       mov    al, [Current_FATType]
0000A13E 3C02                <1>       cmp    al, 2
0000A140 7729                <1>       ja     short loc_reload_FAT_sub_directory
0000A142 8A25[FC580100]      <1>       mov    ah, [Current_Dir_Level]
0000A148 08C0                <1>       or     al, al
0000A14A 740A                <1>       jz     short loc_reload_FS_directory
0000A14C 08E4                <1>       or     ah, ah
0000A14E 751B                <1>       jnz    short loc_reload_FAT_sub_directory
                             <1> loc_reload_FAT_12_16_root_directory:
0000A150 E84E1B0000          <1>       call   load_FAT_root_directory
0000A155 C3                  <1>       retn
                             <1> loc_reload_FS_directory:
0000A156 20E4                <1>       and    ah, ah
0000A158 7506                <1>       jnz    short loc_reload_FS_sub_directory
                             <1> loc_reload_FS_root_directory:
0000A15A E80B1C0000          <1>       call   load_FS_root_directory
0000A15F C3                  <1>       retn
                             <1> loc_reload_FS_sub_directory:
0000A160 A1[F8580100]        <1>       mov    eax, [Current_Dir_FCluster]
0000A165 E8011C0000          <1>       call   load_FS_sub_directory
0000A16A C3                  <1>       retn
                             <1> loc_reload_FAT_sub_directory:
0000A16B A1[F8580100]        <1>       mov    eax, [Current_Dir_FCluster]
0000A170 E8B91B0000          <1>       call   load_FAT_sub_directory
0000A175 C3                  <1>       retn
                             <1>
                             <1> find_directory_entry:
                             <1>       ; 14/02/2016
                             <1>       ; 13/02/2016
                             <1>       ; 10/02/2016
                             <1>       ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 14/08/2010 (DIR.ASM, "proc_find_direntry")
                             <1>       ; 19/09/2009
                             <1>       ; 2005
                             <1>       ; INPUT ->
                             <1>       ;      ESI = Sub Dir or File Name Address
                             <1>       ;      AL = Attributes Mask
                             <1>       ;      (<AL AND EntryAttrib> must be equal to AL)
                             <1>       ;      AH = Negative Attributes Mask (If AH>0)
                             <1>       ;      (<AH AND EntryAttrib> must be ZERO)
                             <1>       ;      CH > 0 Find First Free Dir Entry or Deleted Entry
                             <1>       ;      CL = 0 -> Return the First Free Dir Entry
                             <1>       ;      CL = E5h -> Return the 1st deleted entry
                             <1>       ;      CL = FFh -> Return the 1st deleted or free entry
                             <1>       ;      CL > 0 and CL <> E5h and CL <> FFh -> Return the first
                             <1>       ;             proper entry (which fits with Atributes Masks)
                             <1>       ;      CX = 0 -> Find Valid File/Directory/VolumeName
                             <1>       ;      ? = Any One Char
                             <1>       ;      * = Every Chars
                             <1>       ;      EBX = Current Dir Entry (BX)
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      EDI = Directory Entry Address (in DirectoryBuffer)
                             <1>       ;      ESI = Sub Dir or File Name Address
                             <1>       ;      CF = 0 -> No Error, Proper Entry,
                             <1>       ;      DL = Attributes
                             <1>       ;      DH = Previous Entry Attr (LongName Check)
                             <1>       ;      AL > 0 -> Ambiguous filename wildcard "?" used
                             <1>       ;      AH > 0 -> Ambiguous filename wildcard "*" used
                             <1>       ;      AX = 0 -> Filename full fits with directory entry
                             <1>       ;      EBX = CurrentDirEntry (BX)
                             <1>       ;      CH = The 1st Name Char of Current Dir Entry
                             <1>       ;      CF = 1 -> Proper entry not found, Error Code in AX/AL
                             <1>       ;      CL = 0 and CH = 0 -> Free Entry (End Of Dir)
                             <1>       ;      CL = 0 and CH = E5h -> Deleted Entry fits with filters
                             <1>       ;      CL > 0 -> Entry not found, CH invalid
                             <1>       ;
                             <1>       ; (EAX, EBX, ECX, EDX, EDI, EBP will be changed)
                             <1>
0000A176 663B1D[2B610100]    <1>       cmp    bx, [DirBuff_LastEntry]
0000A17D 0F8739010000        <1>         ja      loc_ffde_stc_retn_255
                             <1>
                             <1>       ;mov    [DirBuff_CurrentEntry], bx
                             <1>
0000A183 BF00000800          <1>       mov    edi, Directory_Buffer
0000A188 66A3[F8610100]      <1>       mov    [FDE_AttrMask], ax
                             <1>
0000A18E 29C0                <1>       sub    eax, eax
                             <1>
                             <1>       ;;mov  [PreviousAttr], al ; 0 ;; 13/02/2016
0000A190 66A3[FA610100]      <1>       mov    [AmbiguousFileName], ax ; 0
                             <1>
0000A196 6689D8              <1>       mov    ax, bx
0000A199 66C1E005            <1>       shl    ax, 5 ; ; * 32 ; Directory entry size
0000A19D 01C7                <1>       add     edi, eax
                             <1>
0000A19F 08ED                <1>       or     ch, ch
0000A1A1 0F852C010000        <1>         jnz     loc_find_free_deleted_entry_0
                             <1>
0000A1A7 08C9                <1>       or      cl, cl
0000A1A9 0F850D010000        <1>         jnz     loc_ffde_stc_retn_255
                             <1>
                             <1> check_find_dir_entry:
0000A1AF 66A1[F8610100]      <1>       mov    ax, [FDE_AttrMask]
0000A1B5 8A2F                <1>       mov    ch, [edi]
0000A1B7 80FD00              <1>       cmp     ch, 0 ; Is it never used entry?
0000A1BA 0F86FF000000        <1>       jna    loc_find_direntry_stc_retn
0000A1C0 56                  <1>       push   esi
0000A1C1 8A570B              <1>       mov    dl, [edi+0Bh] ; File attributes
0000A1C4 80FDE5              <1>       cmp    ch, 0E5h ; Is it a deleted file?
0000A1C7 746D                <1>       je     short loc_find_dir_next_entry_prevdeleted
                             <1>
0000A1C9 80FA0F              <1>       cmp     dl, 0Fh ; longname sub component check
0000A1CC 7505                <1>       jne     short loc_check_attributes_mask
0000A1CE E8ED010000          <1>       call   save_longname_sub_component
                             <1>
                             <1> loc_check_attributes_mask:
0000A1D3 88C6                <1>       mov    dh, al
0000A1D5 20D6                <1>       and    dh, dl
0000A1D7 38F0                <1>       cmp    al, dh
0000A1D9 0F85BA000000        <1>         jne     loc_find_dir_next_entry
0000A1DF 20D4                <1>       and    ah, dl
0000A1E1 0F85B2000000        <1>         jnz     loc_find_dir_next_entry
0000A1E7 80FA0F              <1>       cmp    dl, 0Fh
0000A1EA 751A                <1>       jne    short pass_direntry_attr_check
                             <1>
0000A1EC 3C0F                <1>       cmp    al, 0Fh ; AL = 0Fh -> find long name
0000A1EE 0F85A5000000        <1>         jne     loc_find_dir_next_entry
                             <1>
0000A1F4 5E                  <1>       pop    esi
0000A1F5 6631C0              <1>       xor    ax, ax
0000A1F8 8A35[FC610100]      <1>       mov    dh, [PreviousAttr]
0000A1FE 66891D[29610100]    <1>       mov    [DirBuff_CurrentEntry], bx
0000A205 C3                  <1>       retn
                             <1>
                             <1> pass_direntry_attr_check:
0000A206 89FD                <1>       mov    ebp, edi ; 14/02/2016
0000A208 B908000000          <1>       mov    ecx, 8
                             <1> loc_lodsb_find_dir:
0000A20D AC                  <1>       lodsb
0000A20E 3C2A                <1>       cmp    al, '*'
0000A210 7508                <1>       jne    short pass_fde_ambiguous1_check
0000A212 FE05[FB610100]      <1>         inc     byte [AmbiguousFileName+1]
0000A218 EB28                <1>       jmp    short loc_check_direntry_extension
                             <1>
                             <1> pass_fde_ambiguous1_check:
0000A21A 3C3F                <1>       cmp    al, '?'
0000A21C 750D                <1>       jne    short pass_fde_ambiguous2_check
0000A21E FE05[FA610100]      <1>       inc    byte [AmbiguousFileName]
0000A224 803F20              <1>       cmp    byte [edi], 20h
0000A227 764E                <1>       jna    short loc_find_dir_next_entry_ebp
0000A229 EB14                <1>       jmp    short loc_scasb_find_dir_inc_di
                             <1>
                             <1> pass_fde_ambiguous2_check:
0000A22B 3C20                <1>       cmp    al, 20h
0000A22D 750C                <1>       jne    short loc_scasb_find_dir
0000A22F 803F20              <1>       cmp    byte [edi], 20h
0000A232 7543                <1>       jne    short loc_find_dir_next_entry_ebp
0000A234 EB0C                <1>       jmp    short loc_check_direntry_extension
                             <1>
                             <1> loc_find_dir_next_entry_prevdeleted:
0000A236 80CA80              <1>       or     dl, 80h  ; Bit 7 -> deleted entry sign
0000A239 EB5E                <1>       jmp    short loc_find_dir_next_entry
                             <1>
                             <1> loc_scasb_find_dir:
0000A23B 3A07                <1>       cmp    al, [edi]
0000A23D 7538                <1>       jne    short loc_find_dir_next_entry_ebp
                             <1> loc_scasb_find_dir_inc_di:
0000A23F 47                  <1>       inc    edi
0000A240 E2CB                <1>       loop   loc_lodsb_find_dir
                             <1>
                             <1> loc_check_direntry_extension:
0000A242 BE08000000          <1>       mov    esi, 8
0000A247 89F7                <1>       mov    edi, esi ; 8
0000A249 033424              <1>       add    esi, [esp] ; Sub Dir or File Name Address
0000A24C 01EF                <1>       add    edi, ebp
0000A24E B103                <1>       mov    cl, 3
                             <1> loc_lodsb_find_dir_ext:
0000A250 AC                  <1>       lodsb
0000A251 3C2A                <1>       cmp    al, '*'
0000A253 7508                <1>       jne    short pass_fde_ambiguous3_check
0000A255 FE05[FB610100]      <1>       inc    byte [AmbiguousFileName+1]
0000A25B EB1E                <1>       jmp    short loc_find_dir_proper_direntry
                             <1>
                             <1> pass_fde_ambiguous3_check:
0000A25D 3C3F                <1>       cmp    al, '?'
```

```
0000A25F 750D                <1>       jne    short pass_fde_ambiguous4_check
0000A261 FE05[FA610100]      <1>       inc    byte [AmbiguousFileName]
0000A267 803F20              <1>       cmp    byte [edi], 20h
0000A26A 760B                <1>       jna    short loc_find_dir_next_entry_ebp
0000A26C EB49                <1>       jmp    short loc_scasb_find_dir_ext_inc_di
                             <1>
                             <1> pass_fde_ambiguous4_check:
0000A26E 3C20                <1>       cmp    al, 20h
0000A270 7541                <1>       jne    short loc_scasb_find_dir_ext
0000A272 803F20              <1>       cmp    byte [edi], 20h
0000A275 7404                <1>       je     short loc_find_dir_proper_direntry
                             <1>
                             <1> loc_find_dir_next_entry_ebp:
0000A277 89EF                <1>       mov    edi, ebp ; 14/02/2016
0000A279 EB1E                <1>       jmp    short loc_find_dir_next_entry
                             <1>
                             <1> loc_find_dir_proper_direntry:
0000A27B 30C9                <1>       xor    cl, cl
                             <1> loc_find_dir_proper_direntry_1:
0000A27D 5E                  <1>       pop    esi
0000A27E 89EF                <1>         mov     edi, ebp
0000A280 8A2F                <1>       mov    ch, [edi]
0000A282 8A570B              <1>       mov     dl, [edi+0Bh] ; Dir entry attributes
0000A285 66A1[FA610100]      <1>       mov    ax, [AmbiguousFileName]
                             <1> loc_find_dir_proper_direntry_2:
0000A28B 8A35[FC610100]      <1>       mov     dh, [PreviousAttr]
0000A291 66891D[29610100]    <1>       mov    [DirBuff_CurrentEntry], bx
0000A298 C3                  <1>       retn
                             <1>
                             <1> loc_find_dir_next_entry:
0000A299 8815[FC610100]      <1>       mov    byte [PreviousAttr], dl ; LongName check
                             <1> loc_find_dir_next_entry_1:
0000A29F 5E                  <1>       pop    esi
0000A2A0 83C720              <1>       add    edi, 32
                             <1>       ;inc   word [DirBuff_EntryCounter]
0000A2A3 6643                <1>       inc    bx
0000A2A5 663B1D[2B610100]    <1>       cmp    bx, [DirBuff_LastEntry]
0000A2AC 770E                <1>       ja     short loc_ffde_stc_retn_255
0000A2AE E9FCFEFFFF          <1>         jmp     check_find_dir_entry
                             <1>
                             <1> loc_scasb_find_dir_ext:
0000A2B3 3A07                <1>       cmp    al, [edi]
0000A2B5 75C0                <1>       jne    short loc_find_dir_next_entry_ebp
                             <1> loc_scasb_find_dir_ext_inc_di:
0000A2B7 47                  <1>       inc    edi
0000A2B8 E296                <1>       loop    loc_lodsb_find_dir_ext
0000A2BA EBC1                <1>       jmp    short loc_find_dir_proper_direntry_1
                             <1>
                             <1> loc_ffde_stc_retn_255:
                             <1>       ;mov   cx, 0FFFFh
0000A2BC 31C9                <1>       xor    ecx, ecx
0000A2BE 49                  <1>       dec    ecx ; 0FFFFFFFFh
                             <1>       ;xor   eax, eax
                             <1> loc_find_direntry_stc_retn:
                             <1> loc_check_ffde_retn_1:
                             <1>       ;mov   ax, 2
0000A2BF B802000000          <1>       mov    eax, 2 ; File Not Found
0000A2C4 8A35[FC610100]      <1>       mov    dh, [PreviousAttr]
0000A2CA 66891D[29610100]    <1>       mov    [DirBuff_CurrentEntry], bx
0000A2D1 F9                  <1>       stc
0000A2D2 C3                  <1>       retn
                             <1>
                             <1> loc_find_free_deleted_entry_0:
0000A2D3 66A1[F8610100]      <1>       mov    ax, [FDE_AttrMask]
0000A2D9 8A2F                <1>       mov    ch, [edi]
0000A2DB 8A570B              <1>       mov    dl, [edi+0Bh] ; File attributes
0000A2DE 08C9                <1>       or     cl, cl
0000A2E0 7407                <1>       jz     short loc_check_ffde_0_repeat
                             <1>       ;cmp   cl, 0E5h
                             <1>       ;je    short pass_loc_check_ffde_0_err
0000A2E2 80F9FF              <1>       cmp    cl, 0FFh
0000A2E5 7432                <1>       je     short loc_find_free_deleted_entry_1
0000A2E7 EB4D                <1>       jmp    short pass_loc_check_ffde_0_err
                             <1>
                             <1> loc_check_ffde_0_repeat:
0000A2E9 08ED                <1>       or     ch, ch
0000A2EB 7511                <1>       jnz    short loc_check_ffde_0_next
                             <1>
                             <1> loc_check_ffde_retn_2:
0000A2ED 6629C0              <1>       sub    ax, ax
0000A2F0 8A35[FC610100]      <1>       mov    dh, [PreviousAttr]
0000A2F6 66891D[29610100]    <1>       mov    [DirBuff_CurrentEntry], bx
0000A2FD C3                  <1>       retn
                             <1>
                             <1> loc_check_ffde_0_next:
0000A2FE 6643                <1>       inc    bx
0000A300 83C720              <1>       add    edi, 32
                             <1>       ;inc   word [DirBuff_EntryCounter]
                             <1>
0000A303 663B1D[2B610100]    <1>         cmp  bx, [DirBuff_LastEntry]
0000A30A 77B0                <1>       ja     short loc_ffde_stc_retn_255
0000A30C 8815[FC610100]      <1>       mov    [PreviousAttr], dl
0000A312 8A2F                <1>       mov    ch, [edi]
0000A314 8A570B              <1>       mov    dl, [edi+0Bh] ; file attributes
0000A317 EBD0                <1>       jmp    short loc_check_ffde_0_repeat
                             <1>
                             <1> loc_find_free_deleted_entry_1:
0000A319 28D2                <1>       sub    dl, dl
                             <1> loc_find_free_deleted_entry_2:
0000A31B 20ED                <1>       and    ch, ch
0000A31D 74CE                <1>       jz     short loc_check_ffde_retn_2
0000A31F 80FDE5              <1>       cmp    ch, 0E5h
0000A322 74C9                <1>       je     short loc_check_ffde_retn_2
```

```
0000A324 6643                <1>        inc    bx
0000A326 83C720              <1>        add    edi, 32
0000A329 663B1D[2B610100]    <1>        cmp    bx, [DirBuff_LastEntry]
0000A330 778A                <1>        ja     short loc_ffde_stc_retn_255
0000A332 8A2F                <1>        mov    ch, [edi]
0000A334 EBE5                <1>        jmp    short loc_find_free_deleted_entry_2
                             <1>
                             <1> pass_loc_check_ffde_0_err:
0000A336 38CD                <1>        cmp    ch, cl
0000A338 741F                <1>        je     short loc_check_ffde_attrib
                             <1>
0000A33A 6643                <1>        inc    bx
0000A33C 83C720              <1>        add    edi, 32
0000A33F 663B1D[2B610100]    <1>        cmp    bx, [DirBuff_LastEntry]
0000A346 0F8770FFFFFF        <1>          ja      loc_ffde_stc_retn_255
0000A34C 8815[FC610100]      <1>        mov    [PreviousAttr], dl
0000A352 8A2F                <1>        mov    ch, [edi]
0000A354 8A570B              <1>        mov    dl, [edi+0Bh]
0000A357 EBDD                <1>        jmp    short pass_loc_check_ffde_0_err
                             <1>
                             <1> loc_check_ffde_attrib:
0000A359 88C6                <1>        mov    dh, al
0000A35B 20D6                <1>        and    dh, dl
0000A35D 38F0                <1>        cmp    al, dh
0000A35F 759D                <1>        jne    short loc_check_ffde_0_next
0000A361 20D4                <1>        and    ah, dl
0000A363 7599                <1>        jnz    short loc_check_ffde_0_next
0000A365 30C9                <1>        xor    cl, cl
0000A367 EB84                <1>          jmp     loc_check_ffde_retn_2
                             <1>
                             <1> convert_file_name:
                             <1>        ; 06/03/2016
                             <1>        ; 11/02/2016
                             <1>        ; 07/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>        ; 06/10/2009
                             <1>        ; 2005
                             <1>        ;
                             <1>        ; INPUT  ->
                             <1>        ;      ESI = Dot File Name Location
                             <1>        ;      EDI = Dir Entry Format File Name Location
                             <1>        ; OUTPUT ->
                             <1>        ;      EDI = Dir Entry Format File Name Location
                             <1>        ;      ESI = Dot File Name Location (capitalized)
                             <1>        ;
                             <1>        ; (ECX, AL will be changed)
                             <1>
0000A369 56                  <1>        push   esi
0000A36A 57                  <1>        push   edi
                             <1>
0000A36B B90B000000          <1>        mov    ecx, 11
0000A370 B020                <1>        mov    al, 20h
0000A372 F3AA                <1>        rep    stosb
                             <1>
0000A374 8B3C24              <1>        mov    edi, [esp]
                             <1>
0000A377 B10C                <1>        mov    cl, 12 ; file name length (max.)
                             <1>        ; 06/03/2016
                             <1>        ; Directory entry name limit (11 bytes) check for
                             <1>        ; 'rename_directory_entry' procedure.
                             <1>        ; (EDI points to Directory Entry)
                             <1>        ; (If the file name would not contain a dot
                             <1>        ; and file name length would be 12, this would cause to
                             <1>        ; overwrite the attributes byte of the directory entry.)
                             <1>        ;
0000A379 B50B                <1>        mov    ch, 11 ; directory entry's name length
                             <1> loc_check_first_dot:
0000A37B 8A06                <1>        mov    al, [esi]
0000A37D 3C2E                <1>        cmp    al, 2Eh
0000A37F 750C                <1>        jne    short pass_check_first_dot
0000A381 8807                <1>        mov    [edi], al
0000A383 47                  <1>        inc    edi
0000A384 46                  <1>        inc    esi
0000A385 FEC9                <1>        dec    cl
0000A387 75F2                <1>        jnz    short loc_check_first_dot
                             <1>        ;;(ecx <= 12)
                             <1>        ;;loop loc_check_first_dot
0000A389 EB30                <1>        jmp    short stop_convert_file
                             <1>
                             <1> loc_get_fchar:
0000A38B 8A06                <1>        mov    al, [esi]
                             <1> pass_check_first_dot:
0000A38D 3C61                <1>        cmp    al, 61h ; 'a'
0000A38F 7208                <1>        jb     short pass_name_capitalize
0000A391 3C7A                <1>        cmp    al, 7Ah ; 'z'
0000A393 7704                <1>        ja     short pass_name_capitalize
0000A395 24DF                <1>        and    al, 0DFh
0000A397 8806                <1>        mov    [esi], al
                             <1> pass_name_capitalize:
0000A399 3C21                <1>        cmp    al, 21h
0000A39B 721E                <1>        jb     short stop_convert_file
0000A39D 3C2E                <1>        cmp    al, 2Eh ; '.'
0000A39F 750C                <1>        jne    short pass_dot_space
                             <1> add_dot_space:
0000A3A1 80F904              <1>        cmp    cl, 4
0000A3A4 760E                <1>        jna    short inc_and_loop
0000A3A6 47                  <1>        inc    edi
0000A3A7 FECD                <1>        dec    ch ; 06/03/2016
0000A3A9 FEC9                <1>        dec    cl
0000A3AB EBF4                <1>        jmp    short add_dot_space
                             <1>
                             <1>        ;mov   al, 4
                             <1>        ;cmp   cl, al
                             <1>        ;jna   short inc_and_loop
```

```
                              <1>       ;sub   cl, al
                              <1>       ;add   edi, ecx
                              <1>       ;mov   cl, al
                              <1>       ;jmp   short inc_and_loop
                              <1> 
                              <1> pass_dot_space:
0000A3AD 8807                 <1>       mov    [edi], al
                              <1> loc_after_double_dot:
                              <1>       ; 06/03/2016
0000A3AF FECD                 <1>       dec    ch ; count down for 11 bytes dir entry limit
0000A3B1 740A                 <1>       jz     short stop_convert_file_x
0000A3B3 47                   <1>       inc    edi
                              <1> inc_and_loop:
0000A3B4 FEC9                 <1>       dec    cl ; count down for 12 bytes filename limit
0000A3B6 7403                 <1>       jz     short stop_convert_file
0000A3B8 46                   <1>       inc    esi
                              <1>       ;;(ecx <= 12)
                              <1>       ;;loop loc_get_fchar
0000A3B9 EBD0                 <1>       jmp    short loc_get_fchar
                              <1> 
                              <1> stop_convert_file:
                              <1>       ; 06/03/2016
0000A3BB 30ED                 <1>       xor    ch, ch
                              <1>       ; ECX < 256 ; 'find_first_file' -> xor cl, cl
                              <1> stop_convert_file_x:
0000A3BD 5F                   <1>       pop    edi
0000A3BE 5E                   <1>       pop    esi
0000A3BF C3                   <1>       retn
                              <1> 
                              <1> save_longname_sub_component:
                              <1>       ; 13/02/2016
                              <1>       ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
                              <1>       ; 28/02/2010
                              <1>       ; 17/10/2009
                              <1>       ; INPUT ->
                              <1>       ;      EDI = Directory Entry
                              <1>       ;      // This procedure is called
                              <1>       ;      // from 'find_directory_entry' procedure.
                              <1>       ;      // If the last entry returns with
                              <1>       ;      // a non-zero LongnameFound value and
                              <1>       ;      // if LFN_CheckSum value is equal to
                              <1>       ;      // the next shortname checksum,
                              <1>       ;      // long name is valid.
                              <1>       ;      // If a longname is longer than 65 bytes,
                              <1>       ;      // it is invalid for trdos. (>45h)
                              <1> 
0000A3C0 57                   <1>       push   edi
0000A3C1 56                   <1>       push   esi
                              <1>       ;push  ebx
                              <1>       ;push  ecx
                              <1>       ;push  edx
0000A3C2 50                   <1>       push   eax
                              <1> 
0000A3C3 29C9                 <1>       sub    ecx, ecx
                              <1>       ;sub   eax, eax
0000A3C5 B11A                 <1>       mov    cl, 26
                              <1> 
0000A3C7 0FB607               <1>       movzx  eax, byte [edi] ; LDIR_Order
0000A3CA 3C41                 <1>       cmp    al, 41h  ; 40h (last long entry sign) + 1
0000A3CC 722B                 <1>       jb     short pass_pslnsc_last_long_entry
                              <1> 
0000A3CE 88C4                 <1>       mov    ah, al
0000A3D0 80EC40               <1>       sub    ah, 40h
0000A3D3 8825[FE610100]       <1>       mov    [LFN_EntryLength], ah
                              <1> 
0000A3D9 3C45                 <1>       cmp    al, 45h  ; 40h (last long entry sign) + 5
                              <1>              ; Max 130 byte length is usable in TRDOS
                              <1> ; 26*5 = 130
0000A3DB 7753                 <1>       ja     short loc_pslnsc_retn
                              <1> 
0000A3DD 2407                 <1>       and    al, 07h ; 0Fh
0000A3DF A2[FD610100]         <1>       mov    [LongNameFound], al
                              <1> 
0000A3E4 FEC8                 <1>       dec    al
                              <1>       ;mov   cl, 26
0000A3E6 F6E1                 <1>       mul    cl
                              <1> 
0000A3E8 89C6                 <1>       mov    esi, eax
0000A3EA 01CE                 <1>       add    esi, ecx
                              <1>              ; to make is an ASCIIZ string
                              <1>              ; with ax+26 bytes length
0000A3EC 81C6[00620100]       <1>       add    esi, LongFileName
0000A3F2 66C7060000           <1>       mov    word [esi], 0
0000A3F7 EB16                 <1>       jmp    short loc_pslsc_move_ldir_name2
                              <1> 
                              <1> pass_pslnsc_last_long_entry:
0000A3F9 3C04                 <1>       cmp    al, 04h
0000A3FB 7733                 <1>       ja     short loc_pslnsc_retn
0000A3FD FE0D[FD610100]       <1>       dec    byte [LongNameFound]
0000A403 3A05[FD610100]       <1>       cmp    al, [LongNameFound]
0000A409 7525                 <1>       jne    short loc_pslnsc_retn
                              <1> 
                              <1> loc_pslsc_move_ldir_name1:
0000A40B FEC8                 <1>       dec    al
                              <1>       ;mov   cl, 26
0000A40D F6E1                 <1>       mul    cl
                              <1> 
                              <1> loc_pslsc_move_ldir_name2:
0000A40F 8A4F0D               <1>       mov    cl, [edi+0Dh] ; long name checksum
0000A412 880D[FF610100]       <1>       mov    [LFN_CheckSum], cl
0000A418 89FE                 <1>       mov    esi, edi ; LDIR_Order
0000A41A BF[00620100]         <1>       mov    edi, LongFileName
0000A41F 01C7                 <1>       add    edi, eax
```

```
0000A421 46                  <1>       inc    esi
0000A422 B105                <1>       mov    cl, 5 ; chars 1 to 5
0000A424 F366A5              <1>       rep    movsw
0000A427 83C603              <1>       add    esi, 3
0000A42A A5                  <1>       movsd  ; char 6 & 7
0000A42B A5                  <1>       movsd  ; char 8 & 9
0000A42C A5                  <1>       movsd  ; char 10 & 11
0000A42D 46                  <1>       inc    esi
0000A42E 46                  <1>       inc    esi
0000A42F A5                  <1>       movsd   ; char 12 & 13
                             <1>
                             <1> loc_pslnsc_retn:
0000A430 58                  <1>       pop    eax
                             <1>       ;pop   edx
                             <1>       ;pop   ecx
                             <1>       ;pop   ebx
0000A431 5E                  <1>       pop    esi
0000A432 5F                  <1>       pop    edi
                             <1>
0000A433 C3                  <1>       retn
                             <1>
                             <1> parse_path_name:
                             <1>       ; 10/02/2016
                             <1>       ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 10/009/2011 ('proc_parse_pathname')
                             <1>       ; 27/11/2009
                             <1>       ; 05/12/2004
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      ESI = Beginning of ASCIIZ pathname string
                             <1>       ;       EDI = Destination Address
                             <1>       ;            (which is TR-DOS FindFile data buffer)
                             <1>       ; OUTPUT ->
                             <1>       ;      CF = 1 -> Error
                             <1>       ;           EAX = Error Code (AL)
                             <1>       ;
                             <1>       ; (Modified registers: eax, ecx, esi, edi)
                             <1>
                             <1>       ; Clear the pathname bytes in TR-DOS Findfile data buffer
0000A434 57                  <1>       push   edi
0000A435 B914000000          <1>       mov    ecx, 20  ; 80 bytes
0000A43A 31C0                <1>       xor    eax, eax
0000A43C F3AB                <1>       rep    stosd
0000A43E 5F                  <1>       pop    edi
                             <1>
0000A43F 668B06              <1>       mov    ax, [esi]
0000A442 80FC3A              <1>       cmp    ah, ':'
0000A445 741C                <1>       je     short loc_ppn_change_drive
0000A447 A0[FE580100]        <1>       mov    al, [Current_Drv]
0000A44C EB33                <1>       jmp    short pass_ppn_change_drive
                             <1>
                             <1> pass_ppn_cdir:
0000A44E 8B35[22630100]      <1>       mov    esi, [First_Path_Pos]
0000A454 AC                  <1>       lodsb
                             <1> loc_ppn_get_filename:
0000A455 83C741              <1>       add    edi, 65 ; FindFile_Name location
                             <1>       ; TRDOS Filename length must not be more than 12 bytes
                             <1>       ;mov   ecx, 12
0000A458 B10C                <1>       mov    cl, 12
                             <1> loc_ppn_get_fnchar_next:
0000A45A AA                  <1>       stosb
0000A45B AC                  <1>       lodsb
0000A45C 3C21                <1>       cmp    al, 21h
0000A45E 7274                <1>       jb     short loc_ppn_clc_return
0000A460 E2F8                <1>         loop     loc_ppn_get_fnchar_next
                             <1> loc_ppn_return:
0000A462 C3                  <1>       retn
                             <1>
                             <1> loc_ppn_change_drive:
0000A463 24DF                <1>       and    al, 0DFh
0000A465 2C41                <1>       sub    al, 'A'; A:
0000A467 726F                <1>       jc     short loc_ppn_invalid_drive
0000A469 3805[D20C0100]      <1>       cmp    [Last_DOS_DiskNo], al
0000A46F 7267                <1>       jb     short loc_ppn_invalid_drive
                             <1>
0000A471 46                  <1>       inc    esi
0000A472 46                  <1>       inc    esi
0000A473 8A26                <1>       mov    ah, [esi]
0000A475 80FC21              <1>       cmp    ah, 21h
0000A478 7307                <1>       jnb    short pass_ppn_change_drive
                             <1>
                             <1> loc_ppn_cmd_failed:
                             <1>       ; File or directory name is not existing
0000A47A 8807                <1>       mov    [edi], al ; Drv
0000A47C 66B80100            <1>       mov    ax, 1 ; eax = 1
                             <1>       ; TR-DOS Error Code 01h = Bad Command Argument
                             <1>       ; MS-DOS Error Code 01h : Invalid Function Number
                             <1>       ;stc
                             <1>       ; (MainProg ErrMsg: "Bad command or file name!")
0000A480 C3                  <1>       retn
                             <1>
                             <1> pass_ppn_change_drive:
0000A481 8935[22630100]      <1>       mov    [First_Path_Pos], esi
0000A487 C705[26630100]0000- <1>       mov    dword [Last_Slash_Pos], 0
0000A48F 0000                <1>
0000A491 AA                  <1>       stosb
0000A492 8A06                <1>       mov    al, [esi]
                             <1> loc_scan_ppn_dslash:
0000A494 3C2F                <1>       cmp    al, '/'
0000A496 7506                <1>       jne    short loc_scan_next_slash_pos
0000A498 8935[26630100]      <1>       mov    [Last_Slash_Pos], esi
                             <1> loc_scan_next_slash_pos:
0000A49E 46                  <1>       inc    esi
```

```
0000A49F 8A06                <1>       mov   al, [esi]
0000A4A1 3C20                <1>       cmp   al, 20h
0000A4A3 77EF                <1>       ja    short loc_scan_ppn_dslash
0000A4A5 833D[26630100]00    <1>       cmp   dword [Last_Slash_Pos], 0
0000A4AC 76A0                <1>       jna   short pass_ppn_cdir
                             <1> 
0000A4AE 8B0D[26630100]      <1>       mov   ecx, [Last_Slash_Pos]
0000A4B4 8B35[22630100]      <1>       mov   esi, [First_Path_Pos]
0000A4BA 29F1                <1>       sub   ecx, esi
0000A4BC 41                  <1>       inc   ecx
                             <1>       ;cmp  ecx, 64
0000A4BD 80F940              <1>       cmp   cl, 64
0000A4C0 7715                <1>       ja    short loc_ppn_invalid_drive_stc
                             <1> 
0000A4C2 89F8                <1>       mov   eax, edi ; Dest Dir String Location (65 byte)
0000A4C4 F3A4                <1>       rep   movsb
                             <1>       ;mov  [edi], cl ; 0, End of Dir String
0000A4C6 8B35[26630100]      <1>       mov   esi, [Last_Slash_Pos]
0000A4CC 46                  <1>       inc   esi
0000A4CD 89C7                <1>       mov   edi, eax
0000A4CF AC                  <1>       lodsb
0000A4D0 3C21                <1>       cmp   al, 21h
0000A4D2 7381                <1>       jnb   short loc_ppn_get_filename
                             <1> loc_ppn_clc_return:
                             <1>       ;clc
0000A4D4 31C0                <1>       xor   eax, eax
0000A4D6 C3                  <1>       retn
                             <1> 
                             <1> loc_ppn_invalid_drive_stc:
0000A4D7 F5                  <1>       cmc     ; stc
                             <1> loc_ppn_invalid_drive:
                             <1>       ; cf = 1
                             <1>       ; The Drive Letter/Char < "A" or > "Z"
0000A4D8 66B80F00            <1>       mov   ax, 0Fh
                             <1>       ; MS-DOS Error Code 0Fh = Disk Drive Invalid
                             <1>       ; (MainProg ErrMsg: "Drive not ready or read error!")
0000A4DC C3                  <1>       retn
                             <1> 
                             <1> find_longname:
                             <1>       ; 13/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 24/01/2010 (DIR.ASM, 'proc_find_longname')
                             <1>       ; 17/10/2009
                             <1> 
                             <1>       ; INPUT ->
                             <1>       ;     ESI = DOS short file name address
                             <1>       ;     for example: "filename.ext"
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;     ESI = ASCIIZ longname address (cf = 0)
                             <1>       ;     cf = 1 -> error number returns in EAX (AL)
                             <1>       ;     AL = 0 & CF=1 -> longname not found
                             <1>       ;          the file/directory has no longname
                             <1>       ;     cf = 0 -> AL = FAT Type
                             <1> 
                             <1>       ; 17/10/2009
                             <1>       ; ASCIIZ string will be returned
                             <1>       ; as LongFileName
                             <1>       ; clearing/reset is not needed
                             <1>       ;mov  ecx, 33
                             <1>       ;mov  edi, LongFileName
                             <1>       ;sub  ax, ax ; 0
                             <1>       ;rep  stosw
                             <1> 
                             <1>       ;mov  byte [LongNameFound], 0
                             <1> 
                             <1>       ; ESI = ASCIIZ file/directory name address
                             <1>       ;   AL = Attributes AND mask
                             <1>       ;     (Result of AND must be equal to AL)
                             <1>       ;   AH = Negative attributes mask
                             <1>       ;     (Result of AND must be ZERO)
0000A4DD 66B80008            <1>       mov   ax, 0800h
                             <1>             ; it must not be volume name or longname
0000A4E1 E87DDDFFFF          <1>       call  find_first_file
0000A4E6 7216                <1>       jc    short loc_fln_retn
                             <1> 
                             <1> loc_fln_check_FAT_Type:
0000A4E8 803D[FD580100]01    <1>       cmp   byte [Current_FATType], 1
0000A4EF 7306                <1>       jnb   short loc_fln_check_longname_yes_sign
                             <1> 
0000A4F1 E839000000          <1>       call  get_fs_longname
0000A4F6 C3                  <1>       retn
                             <1> 
                             <1> loc_fln_check_longname_yes_sign:
0000A4F7 08FF                <1>       or    bh, bh
0000A4F9 7504                <1>       jnz   short loc_fln_check_longnamefound_number
                             <1> loc_fln_longname_not_found_retn:
0000A4FB 31C0                <1>       xor   eax, eax
                             <1>       ; cf = 1 & al = 0 -> longname not found
0000A4FD F9                  <1>       stc
                             <1> loc_fln_retn:
0000A4FE C3                  <1>       retn
                             <1> 
                             <1> loc_fln_check_longnamefound_number:
                             <1>       ; 'LongNameFound' is set by
                             <1>         ; by 'save_longname_sub_component'
                             <1>       ; which is called from
                             <1>       ; 'find_directory_entry'
                             <1>       ; which is called from
                             <1>       ; 'find_first_file'
                             <1>       ; It must 1 if the longname is valid
0000A4FF 803D[FD610100]01    <1>         cmp     byte [LongNameFound], 1
0000A506 75F3                <1>       jne   short loc_fln_longname_not_found_retn
                             <1> 
                             <1> loc_fln_calculate_checksum:
0000A508 E813000000          <1>       call   calculate_checksum
                             <1>       ; AL = shortname checksum
                             <1>
                             <1> loc_fln_longname_validation:
                             <1>       ; 'LFN_CheckSum' has been set already
                             <1>       ; by 'save_longname_sub_component'
                             <1>       ; which is called from
                             <1>       ; 'find_directory_entry'
                             <1>       ; which is called from
                             <1>       ; 'find_first_file'
0000A50D 3805[FF610100]      <1>       cmp    [LFN_CheckSum], al
0000A513 75E6                <1>       jne    short loc_fln_longname_not_found_retn
                             <1>
0000A515 BE[00620100]        <1>       mov    esi, LongFileName
0000A51A A0[FD580100]        <1>       mov    al, [Current_FATType]
0000A51F C3                  <1>       retn
                             <1>
                             <1> calculate_checksum:
                             <1>       ; 13/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 17/10/2009 (DIR.ASM, 'proc_calculate_checksum')
                             <1>         ;
                             <1>       ; INPUT ->
                             <1>       ;      ESI = 11 byte DOS File Name location
                             <1>       ;      (in DOS Directory Entry Format)
                             <1>       ; OUTPUT ->
                             <1>       ;       AL = 8 bit checksum (CRC) value
                             <1>       ;
                             <1>       ; (Modified registers: EAX, ECX, ESI)
                             <1>
                             <1>       ; Erdogan Tan [ 17-10-2009 ]
                             <1>       ;  'ror al, 1' instruction
                             <1>
                             <1>       ; Erdogan Tan [ 20-06-2004 ]
                             <1>       ; This 8086 assembly code is an original code
                             <1>       ; which is adapted from C code in
                             <1>       ; Microsoft FAT32 File System Specification
                             <1>       ; Version 1.03, December 6, 2000
                             <1>       ; Page 28
                             <1>
0000A520 30C0                <1>       xor    al, al
0000A522 B90B000000          <1>       mov    ecx, 11
                             <1> loc_next_sum:
                             <1>       ;xor   ah, ah
                             <1>       ;test  al, 1
                             <1>       ;jz    short pass_ah_80h
                             <1>       ;mov   ah, 80h
                             <1> ;pass_ah_80h:
                             <1>       ;shr   al, 1
0000A527 D0C8                <1>       ror    al, 1 ; 17/10/2009
0000A529 0206                <1>       add    al, [esi]
0000A52B 46                  <1>       inc    esi
                             <1>       ;add   al, ah
0000A52C E2F9                <1>       loop   loc_next_sum
0000A52E C3                  <1>       retn
                             <1>
                             <1> get_fs_longname:
                             <1>       ; temporary (13/02/2016)
0000A52F 31C0                <1>       xor eax, eax
0000A531 F9                  <1>       stc
0000A532 C3                  <1>       retn
                             <1>
                             <1> make_sub_directory:
                             <1>       ; 16/10/2016
                             <1>       ; 02/03/2016, 03/03/2016
                             <1>       ; 26/02/2016, 27/02/2016
                             <1>       ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 01/08/2011 (DIR.ASM, 'proc_make_directory')
                             <1>       ; 10/07/2010
                             <1>       ; INPUT ->
                             <1>       ;      ESI = ASCIIZ Directory Name
                             <1>       ;      CL = Directory Attributes
                             <1>       ; OUTPUT ->
                             <1>       ;      EAX = New sub dir's first cluster
                             <1>       ;      ESI = Logical Dos Drv Descr. Table Addr.
                             <1>       ;      CF = 1 -> error code in AL (EAX)
                             <1>
                             <1>       ;test  cl, 10h  ; directory
                             <1>       ;jz    short loc_make_directory_access_denied
                             <1>       ;test  cl, 08h ; volume name
                             <1>       ;jnz   short loc_make_directory_access_denied
                             <1>
0000A533 80E107              <1>       and    cl, 07h
0000A536 880D[7C630100]      <1>       mov    byte [mkdir_attrib], cl
                             <1>
0000A53C 56                  <1>       push   esi
0000A53D 31DB                <1>       xor    ebx, ebx
0000A53F 8A3D[FE580100]      <1>       mov    bh, [Current_Drv]
0000A545 BE00010900          <1>       mov    esi, Logical_DOSDisks
0000A54A 01DE                <1>       add    esi, ebx
0000A54C 5B                  <1>       pop    ebx
                             <1>
                             <1>       ; 10/07/2010 -> 1st writable disk check for trdos
                             <1>       ; LD_DiskType = 0 for write protection (read only)
0000A54D 807E0101            <1>       cmp    byte [esi+LD_DiskType], 1 ; 0 = Invalid
0000A551 730B                <1>       jnb    short loc_mkdir_check_file_sytem
                             <1>       ; 16/10/2016 (13h -> 30)
0000A553 B81E000000          <1>       mov    eax, 30 ; 'Disk write-protected' error
0000A558 BA00000000          <1>       mov    edx, 0
                             <1>       ; err retn: EDX = 0, EBX = Dir name offset
                             <1>       ;ESI = Logical DOS drive description table address
0000A55D C3                  <1>       retn
                             <1>
```

```
                              <1> ;loc_make_directory_access_denied:
                              <1>       ;mov   ax, 05h ; access denied (invalid attributes input)
                              <1>       ;stc
                              <1>       ;retn
                              <1>
                              <1> loc_mkdir_check_file_sytem:
0000A55E 807E0301             <1>       cmp    byte [esi+LD_FATType], 1
0000A562 730B                 <1>       jnb    short loc_mkdir_check_free_sectors
                              <1>
                              <1> loc_make_fs_directory:
0000A564 A1[F8580100]         <1>       mov    eax, [Current_Dir_FCluster]
                              <1>       ; EAX = Parent directory DDT Address
                              <1>       ; ESI = Logical DOS Drive DT Address
                              <1>       ; EBX = Directory name offset (as ASCIIZ name)
0000A569 E8D5150000           <1>       call   make_fs_directory
0000A56E C3                   <1>       retn
                              <1>
                              <1> loc_mkdir_check_free_sectors:
0000A56F 0FB64613             <1>         movzx   eax, byte [esi+LD_BPB+SecPerClust]
0000A573 8B4E74               <1>       mov    ecx, [esi+LD_FreeSectors]
0000A576 39C1                 <1>       cmp    ecx, eax
0000A578 7255                 <1>       jb     short loc_mkdir_insufficient_disk_space
                              <1>
                              <1> loc_make_fat_directory:
0000A57A 891D[6C630100]       <1>       mov    [mkdir_DirName_Offset], ebx
0000A580 890D[78630100]       <1>       mov    [mkdir_FreeSectors], ecx
                              <1>
                              <1>       ;mov   al, [esi+LD_BPB+SecPerClust]
0000A586 A2[7E630100]         <1>       mov    byte [mkdir_SecPerClust], al
                              <1>
                              <1> loc_mkdir_gffc_1:
0000A58B E80F180000           <1>       call   get_first_free_cluster
0000A590 722A                 <1>       jc     short loc_mkdir_gffc_retn
                              <1>
                              <1> ;loc_mkdir_gffc_1_cont:
                              <1>       ;cmp   eax, 2
                              <1>       ;jb    short loc_mkdir_gffc_insufficient_disk_space
                              <1>
                              <1> ;loc_mkdir_gffc_1_save_fcluster:
0000A592 A3[70630100]         <1>       mov    [mkdir_FFCluster], eax
                              <1>
                              <1> loc_mkdir_locate_ffe:
                              <1>       ; Current directory fcluster <> Directory buffer cluster
                              <1>       ; Current directory will be reloaded by
                              <1>       ; 'locate_current_dir_file' procedure
                              <1>       ;
                              <1>       ; ESI = Logical DOS Drive Description Table Address
                              <1>       ;push  esi ; 27/02/2016
0000A597 31C0                 <1>       xor    eax, eax
0000A599 89C1                 <1>         mov ecx, eax
0000A59B 6649                 <1>       dec    cx ; FFFFh
                              <1>       ; CX = FFFFh -> find first deleted or free entry
                              <1>       ; ESI would be ASCIIZ filename address if the call
                              <1>       ; would not be for first free or deleted dir entry
0000A59D E8D0FAFFFF           <1>       call   locate_current_dir_file
0000A5A2 734C                 <1>       jnc    short loc_mkdir_set_ff_dir_entry_1
                              <1>       ;pop   esi
                              <1>       ; ESI = Logical DOS Drive Description Table Address
0000A5A4 83F802               <1>       cmp    eax, 2  ; cmp al, 2 ; File/Dir not found !
0000A5A7 752B                 <1>       jne    short loc_mkdir_stc_return
                              <1>
                              <1> loc_mkdir_add_new_cluster:
0000A5A9 3805[FD580100]       <1>       cmp    byte [Current_FATType], al ; 2
                              <1>       ;cmp   byte ptr [esi+LD_FATType], 2
0000A5AF 770C                 <1>       ja     short loc_mkdir_add_new_cluster_check_fsc
0000A5B1 803D[FC580100]01     <1>       cmp    byte [Current_Dir_Level], 1
                              <1>       ;cmp   byte [esi+LD_CDirLevel], 1
0000A5B8 7303                 <1>       jnb    short loc_mkdir_add_new_cluster_check_fsc
                              <1>
0000A5BA B00C                 <1>       mov    al, 12 ; No more files
                              <1> loc_mkdir_gffc_retn:
0000A5BC C3                   <1>       retn
                              <1>
                              <1> loc_mkdir_add_new_cluster_check_fsc:
0000A5BD 8B0D[78630100]       <1>       mov    ecx, [mkdir_FreeSectors]
                              <1>       ;movzx eax, byte [mkdir_SecPerClust]
0000A5C3 A0[7E630100]         <1>       mov    al, [mkdir_SecPerClust]
0000A5C8 66D1E0               <1>       shl    ax, 1 ; AX = 2 * AX
0000A5CB 39C1                 <1>       cmp    ecx, eax
0000A5CD 7350                 <1>       jnb    short loc_mkdir_add_new_subdir_cluster
                              <1>
                              <1> loc_mkdir_insufficient_disk_space:
                              <1>       ;mov   edx, ecx
                              <1> ;loc_mkdir_gffc_insufficient_disk_space:
0000A5CF 66B82700             <1>       mov    ax, 27h ; MSDOS err => insufficient disk space
                              <1>       ; err retn: EDX = Free sectors, EBX = Dir name offset
                              <1>         ; ESI -> Dos drive description table address
                              <1>       ;; ecx = edx
                              <1>       ;
0000A5D3 C3                   <1>       retn
                              <1>
                              <1> loc_mkdir_stc_return:
0000A5D4 F9                   <1>       stc
0000A5D5 C3                   <1>       retn
                              <1>
                              <1> loc_mkdir_gffc_2:
0000A5D6 E8C4170000           <1>       call   get_first_free_cluster
0000A5DB 72DF                 <1>       jc     short loc_mkdir_gffc_retn
                              <1>
                              <1> ;loc_mkdir_gffc_1_cont:
                              <1>       ;cmp   eax, 2
                              <1>       ;jb    short loc_mkdir_gffc_insufficient_disk_space
                              <1>
```

```
                                  <1> ;loc_mkdir_gffc_2_save_fcluster:
0000A5DD A3[70630100]             <1>       mov    [mkdir_FFCluster], eax
                                  <1>
0000A5E2 A1[74630100]             <1>       mov    eax, [mkdir_LastDirCluster]
                                  <1>
0000A5E7 E842170000               <1>       call   load_FAT_sub_directory
0000A5EC 72CE                     <1>       jc     short loc_mkdir_gffc_retn
                                  <1>
0000A5EE 31FF                     <1>       xor    edi, edi
                                  <1> loc_mkdir_set_ff_dir_entry_1:
                                  <1>       ; 27/02/2016
0000A5F0 56                       <1>       push   esi ; Logical DOS Drv Desc. Tbl. address
                                  <1>       ; EDI = Directory Entry Address
0000A5F1 8B35[6C630100]           <1>       mov    esi, [mkdir_DirName_Offset]
0000A5F7 A1[70630100]             <1>       mov    eax, [mkdir_FFCluster]
                                  <1>
0000A5FC 66B91000                 <1>       mov    cx, 10h      ; CL = Directory attribute
                                  <1>                     ; CH = 0 -> File size is 0
0000A600 0A0D[7C630100]           <1>       or     cl, [mkdir_attrib] ; S, H, R
0000A606 E8B0010000               <1>       call   make_directory_entry
                                  <1>
0000A60B 5E                       <1>       pop    esi
                                  <1>
0000A60C C605[28610100]02         <1>       mov    byte [DirBuff_ValidData], 2
0000A613 E880020000               <1>       call   save_directory_buffer
0000A618 0F83DA000000             <1>         jnc     loc_mkdir_set_ff_dir_entry_2
                                  <1>
                                  <1> loc_mkdir_return:
0000A61E C3                       <1>       retn
                                  <1>
                                  <1> loc_mkdir_add_new_subdir_cluster:
0000A61F 8B15[2D610100]           <1>       mov    edx, [DirBuff_Cluster]
0000A625 8915[74630100]           <1>       mov    [mkdir_LastDirCluster], edx
                                  <1>
0000A62B A1[70630100]             <1>       mov    eax, [mkdir_FFCluster]
0000A630 E8F9160000               <1>       call   load_FAT_sub_directory
0000A635 72E7                     <1>       jc     short loc_mkdir_return
                                  <1>       ; eax = 0
                                  <1>       ; ecx =  directory buffer sector count (<= 128)
                                  <1>
                                  <1> pass_mkdir_add_new_subdir_cluster:
0000A637 29FF                     <1>       sub    edi, edi ; 0
                                  <1>       ;mov   al, 128 ; double word
                                  <1>       ;mul   ecx ; ecx =  directory buffer sector count
                                  <1>       ;mov   ecx, eax
                                  <1>       ;shl   cx, 7 ; 128 * sector count
0000A639 668B4611                 <1>       mov    ax, [esi+LD_BPB+BytesPerSec] ; 512
0000A63D 66C1E802                 <1>       shr    ax, 2 ; 'byte count / 4' for 'stosd'
0000A641 66F7E1                   <1>       mul    cx ; max = 128*(512/4) -> 16384 (stosd)
0000A644 6689C1                   <1>       mov    cx, ax
0000A647 6629C0                   <1>       sub    ax, ax ; 0
0000A64A F3AB                     <1>       rep    stosd ; clear directory buffer
                                  <1>
0000A64C C605[28610100]02         <1>       mov    byte [DirBuff_ValidData], 2
0000A653 E840020000               <1>       call   save_directory_buffer
0000A658 72C4                     <1>       jc     short loc_mkdir_return
                                  <1>
                                  <1> loc_mkdir_save_added_cluster:
0000A65A A1[74630100]             <1>       mov    eax, [mkdir_LastDirCluster]
0000A65F 8B0D[70630100]           <1>       mov    ecx, [mkdir_FFCluster]
                                  <1>       ; 01/03/2016
0000A665 31D2                     <1>       xor    edx, edx
0000A667 8915[1E610100]           <1>       mov    [FAT_ClusterCounter], edx ; 0 ; reset
0000A66D E800180000               <1>       call   update_cluster
0000A672 7304                     <1>       jnc    short loc_mkdir_save_fat_buffer_0
0000A674 09C0                     <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
0000A676 7518                     <1>       jnz    short loc_mkdir_save_fat_buffer_stc_retn
                                  <1>
                                  <1> loc_mkdir_save_fat_buffer_0:
0000A678 A1[70630100]             <1>       mov    eax, [mkdir_FFCluster]
0000A67D A3[74630100]             <1>       mov    [mkdir_LastDirCluster], eax
                                  <1>
0000A682 31C9                     <1>       xor    ecx, ecx
0000A684 49                       <1>       dec    ecx ; FFFFFFFFh
                                  <1>       ; ESI = Logical DOS Drive Description Table address
0000A685 E8E8170000               <1>       call   update_cluster
0000A68A 731A                     <1>       jnc    short loc_mkdir_save_fat_buffer_1
0000A68C 09C0                     <1>       or     eax, eax
0000A68E 7416                     <1>       jz     short loc_mkdir_save_fat_buffer_1
                                  <1>
                                  <1> loc_mkdir_save_fat_buffer_stc_retn:
                                  <1>       ; 01/03/2016
0000A690 803D[1E610100]01         <1>       cmp    byte [FAT_ClusterCounter], 1
0000A697 720C                     <1>       jb     short loc_mkdir_save_fat_buffer_retn
                                  <1>
0000A699 66BB00FF                 <1>       mov    bx, 0FF00h ; recalculate free space (BL = 0)
                                  <1>                         ; (BH = FFh -> Use ESI as Drv Param. Tbl.)
0000A69D 50                       <1>       push   eax
0000A69E E8211B0000               <1>       call   calculate_fat_freespace
0000A6A3 58                       <1>       pop    eax
0000A6A4 F9                       <1>       stc
                                  <1> loc_mkdir_save_fat_buffer_retn:
0000A6A5 C3                       <1>       retn
                                  <1>
                                  <1> loc_mkdir_save_fat_buffer_1:
                                  <1>       ; byte [FAT_BuffValidData] = 2
0000A6A6 E8841A0000               <1>       call   save_fat_buffer
0000A6AB 72E3                     <1>       jc     short loc_mkdir_save_fat_buffer_stc_retn
                                  <1>
                                  <1>       ; 01/03/2016
0000A6AD 803D[1E610100]01         <1>       cmp    byte [FAT_ClusterCounter], 1
0000A6B4 721B                     <1>       jb     short loc_mkdir_save_fat_buffer_2
                                  <1>
```

```
                             <1>       ; ESI = Logical DOS Drive Description Table address
0000A6B6 A1[1E610100]        <1>       mov    eax, [FAT_ClusterCounter]
0000A6BB 66BB01FF            <1>       mov    bx, 0FF01h ; add free clusters
0000A6BF E8001B0000          <1>       call   calculate_fat_freespace
                             <1>
                             <1>       ;inc   eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!
                             <1>       ;jnz   short loc_mkdir_save_fat_buffer_2
                             <1>
                             <1>       ; ecx > 0 -> Recalculation is needed
0000A6C4 09C9                <1>       or     ecx, ecx
0000A6C6 7409                <1>       jz     short loc_mkdir_save_fat_buffer_2
                             <1>
0000A6C8 66BB00FF            <1>       mov    bx, 0FF00h ; ; recalculate free space
0000A6CC E8F31A0000          <1>       call   calculate_fat_freespace
                             <1>
                             <1> loc_mkdir_save_fat_buffer_2:
0000A6D1 C605[7F630100]01    <1>       mov    byte [mkdir_add_new_cluster], 1
0000A6D8 E9C4000000          <1>       jmp    loc_mkdir_upd_parent_dir_lmdt
                             <1>
                             <1> loc_mkdir_update_sub_dir_cluster:
0000A6DD A1[70630100]        <1>       mov    eax, [mkdir_FFCluster]
0000A6E2 29C9                <1>       sub    ecx, ecx ; 0
                             <1>       ; 01/03/2016
0000A6E4 890D[1E610100]      <1>       mov    [FAT_ClusterCounter], ecx ; 0 ; Reset
0000A6EA 49                  <1>       dec    ecx ; 0FFFFFFFFh
                             <1>
                             <1>       ; ESI = Logical DOS Drive Descisption Table address
0000A6EB E882170000          <1>       call   update_cluster
0000A6F0 7379                <1>       jnc    short loc_mkdir_save_fat_buffer_3
0000A6F2 09C0                <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
0000A6F4 7475                <1>       jz     short loc_mkdir_save_fat_buffer_3
                             <1>       ; 01/03/2016
0000A6F6 EB98                <1>       jmp    short loc_mkdir_save_fat_buffer_stc_retn
                             <1>
                             <1> loc_mkdir_set_ff_dir_entry_2:
                             <1>       ; ESI = Logical DOS Drive Description Table address
0000A6F8 A1[70630100]        <1>       mov    eax, [mkdir_FFCluster]
                             <1>       ; Load disk sectors as a directory cluster
0000A6FD E82C160000          <1>       call   load_FAT_sub_directory
0000A702 7266                <1>       jc     short retn_make_fat_directory
                             <1>
                             <1>       ; eax = 0
                             <1>       ; ecx =  directory buffer sector count (<= 128)
                             <1>
0000A704 BF40000800          <1>       mov    edi, Directory_Buffer + 64 ; 26/02/2016
                             <1>
                             <1>       ; 02/03/2016
0000A709 668B4611            <1>       mov    ax, [esi+LD_BPB+BytesPerSec] ; 512
0000A70D 66C1E802            <1>       shr    ax, 2 ; 'byte count / 4' for 'stosd'
0000A711 F7E1                <1>       mul    ecx
0000A713 89C1                <1>       mov    ecx, eax
0000A715 6629C0              <1>       sub    ax, ax
0000A718 F3AB                <1>       rep    stosd
                             <1>
                             <1>       ;;mov  al, 128 ; double word
                             <1>       ;;mul  ecx ; ecx =  directory buffer sector count
                             <1>       ;;mov  ecx, eax
                             <1>       ;shl   cx, 7 ; 128 * sector count
                             <1>       ;;sub  eax, eax
                             <1>       ;;sub  al, al ; 0
                             <1>       ;rep   stosd ; clear directory buffer
                             <1>
0000A71A BF00000800          <1>       mov    edi, Directory_Buffer ; 26/02/2016
                             <1>
0000A71F 56                  <1>       push   esi
                             <1>
0000A720 BE[80630100]        <1>       mov    esi, mkdir_Name
0000A725 66C7062E00          <1>       mov    word [esi], 2Eh ; db '.', '0'
                             <1>
0000A72A A1[70630100]        <1>       mov    eax, [mkdir_FFCluster]
0000A72F 66B91000            <1>       mov    cx, 10h ; CL = Directory attribute
                             <1>                    ; CH = 0 -> File size is 0
0000A733 E883000000          <1>       call   make_directory_entry
                             <1>
0000A738 BF20000800          <1>       mov    edi, Directory_Buffer + 32 ; 26/02/2016
                             <1>
                             <1>       ; 03/03/2016
                             <1>       ; Following modification has been done according to
                             <1>       ; 'Microsoft Extensible Firmware Initiative
                             <1>       ; FAT32 File System Specification' document,
                             <1>       ; 'FAT: General Overview of On-Disk Format—Page 25'.
                             <1>       ; "Finally, you set DIR_FstClusLO and DIR_FstClusHI
                             <1>       ; for the dotdot entry (the second entry) to the
                             <1>       ; first cluster number of the directory in which you
                             <1>       ; just created the directory (value is 0 if this directory
                             <1>       ; is the root directory even for FAT32 volumes)."
                             <1>       ; (Correctness of this modification has been verified
                             <1>       ;  by using Windows 98 'scandisk.exe'.)
                             <1>
0000A73D 29C0                <1>       sub    eax, eax
0000A73F 3805[FC580100]      <1>       cmp    byte [Current_Dir_Level], al ; 0
0000A745 7605                <1>       jna    short loc_mkdir_set_ff_dir_entry_3
0000A747 A1[F8580100]        <1>       mov    eax, [Current_Dir_FCluster] ; parent dir
                             <1> loc_mkdir_set_ff_dir_entry_3:
0000A74C 66C746012E00        <1>       mov    word [esi+1], 2Eh ; db '.', '0'
                             <1>
                             <1>       ;mov   cx, 10h
0000A752 E864000000          <1>       call   make_directory_entry
                             <1>
0000A757 5E                  <1>       pop    esi
                             <1>
0000A758 C605[28610100]02    <1>       mov    byte [DirBuff_ValidData], 2
0000A75F E834010000          <1>       call   save_directory_buffer
```

```
0000A764 0F8373FFFFFF        <1>         jnc     loc_mkdir_update_sub_dir_cluster
                             <1> 
                             <1> retn_make_fat_directory:
0000A76A C3                  <1>       retn
                             <1> 
                             <1> loc_mkdir_save_fat_buffer_3:
                             <1>       ; 01/03/2016
                             <1>       ; byte [FAT_BuffValidData] = 2
0000A76B E8BF190000          <1>       call  save_fat_buffer
0000A770 0F821AFFFFFF        <1>         jc      loc_mkdir_save_fat_buffer_stc_retn
                             <1> 
0000A776 803D[1E610100]01    <1>       cmp   byte [FAT_ClusterCounter], 1
0000A77D 721B                <1>       jb    short loc_mkdir_save_fat_buffer_4
                             <1> 
                             <1>       ; ESI = Logical DOS Drive Description Table address
0000A77F A1[1E610100]        <1>       mov   eax, [FAT_ClusterCounter]
0000A784 66BB01FF            <1>       mov   bx, 0FF01h ; add free clusters
0000A788 E8371A0000          <1>       call  calculate_fat_freespace
                             <1> 
                             <1>       ;inc  eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!
                             <1>         ;jnz    short loc_mkdir_save_fat_buffer_4
                             <1> 
                             <1>       ; ecx > 0 -> Recalculation is needed
0000A78D 09C9                <1>       or    ecx, ecx
0000A78F 7409                <1>         jz      short loc_mkdir_save_fat_buffer_4
                             <1> 
0000A791 66BB00FF            <1>       mov   bx, 0FF00h ; recalculate free space
0000A795 E82A1A0000          <1>       call  calculate_fat_freespace
                             <1> 
                             <1> loc_mkdir_save_fat_buffer_4:
0000A79A C605[7F630100]00    <1>       mov   byte [mkdir_add_new_cluster], 0
                             <1> 
                             <1> loc_mkdir_upd_parent_dir_lmdt:
0000A7A1 E88D010000          <1>       call  update_parent_dir_lmdt
                             <1> 
                             <1>       ; 01/03/2016
0000A7A6 803D[7F630100]00    <1>       cmp   byte [mkdir_add_new_cluster], 0
0000A7AD 0F8723FEFFFF        <1>         ja      loc_mkdir_gffc_2
                             <1> 
                             <1> loc_mkdir_retn_new_dir_cluster:
0000A7B3 A1[70630100]        <1>       mov   eax, [mkdir_FFCluster]
0000A7B8 31D2                <1>       xor   edx, edx
                             <1> loc_mkdir_retn:
0000A7BA C3                  <1>       retn
                             <1> 
                             <1> make_directory_entry:
                             <1>       ; 02/03/2016
                             <1>       ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 09/08/2010 (DIR.ASM, 'proc_make_directory_entry')
                             <1>       ; 17/07/2010
                             <1>       ; INPUT ->
                             <1>       ;     EDI = Directory Entry Address
                             <1>       ;     ESI = Dot File Name Location
                             <1>       ;     EAX = First Cluster
                             <1>       ;     File Size = 0 (Must be set later)
                             <1>       ;     CL = Attributes
                             <1>       ;     CH = 0 (File size = 0)
                             <1>       ;     (If CH>0, File size is in dword [EBX]) (*)
                             <1>       ; OUTPUT ->
                             <1>       ;     EDI = Directory Entry Address
                             <1>       ;     ESI = Dot File Name Location (Capitalized)
                             <1>       ;     If CH input = 0, File Size = 0
                             <1>       ;     Otherwise file size is as dword [EBX] (*)
                             <1>       ;     DX = Date, AX = Time in DOS Dir Entry format
                             <1>       ;     EBX = same
                             <1>       ;     ECX = same
                             <1> 
0000A7BB 51                  <1>       push  ecx
                             <1> 
0000A7BC 884F0B              <1>       mov   [edi+11], cl ; Attributes
0000A7BF 6689471A            <1>       mov   [edi+26], ax ; FClusterLw, 26
0000A7C3 C1E810              <1>       shr   eax, 16
0000A7C6 66894714            <1>       mov   [edi+20], ax ; FClusterHw, 20
0000A7CA 6631C0              <1>       xor   ax, ax
0000A7CD 6689470C            <1>       mov   [edi+12], ax ; NTReserved, 12
                             <1>                        ; CrtTimeTenth, 13
0000A7D1 08ED                <1>       or    ch, ch
0000A7D3 7402                <1>       jz    short loc_make_direntry_set_filesize
                             <1> 
0000A7D5 8B03                <1>       mov   eax, [ebx]
                             <1> 
                             <1> loc_make_direntry_set_filesize:
0000A7D7 89471C              <1>       mov   [edi+28], eax ; FileSize, 28
                             <1> 
0000A7DA E88AFBFFFF          <1>       call  convert_file_name
                             <1>       ;EDI = Dir Entry Format File Name Location
                             <1>       ;ESI = Dot File Name Location (capitalized)
                             <1> 
0000A7DF E816000000          <1>       call  convert_current_date_time
                             <1>       ; OUTPUT -> DX = Date in dos dir entry format
                             <1>         ;        AX = Time in dos dir entry format
0000A7E4 6689470E            <1>       mov   [edi+14], ax ; CrtTime, 14
0000A7E8 66895710            <1>       mov   [edi+16], dx ; CrtDate, 16
0000A7EC 66895712            <1>       mov   [edi+18], dx ; LastAccDate, 18
0000A7F0 66894716            <1>       mov   [edi+22], ax ; WrtTime, 14
0000A7F4 66895718            <1>       mov   [edi+24], dx ; WrtDate, 16
0000A7F8 59                  <1>       pop   ecx
                             <1> 
0000A7F9 C3                  <1>       retn
                             <1> 
                             <1> convert_current_date_time:
                             <1>       ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 13/06/2010 (DIR.ASM, 'proc_convert_current_date_time')
```

```
                              <1>       ; converts date&time to dos dir entry format
                              <1>       ; INPUT -> none
                              <1>       ; OUTPUT -> DX = Date in dos dir entry format
                              <1>       ;           AX = Time in dos dir entry format
                              <1>
0000A7FA B404                 <1>       mov    ah, 04h ; Return Current Date
0000A7FC E879B1FFFF           <1>       call   int1Ah
                              <1>
0000A801 88E8                 <1>       mov    al, ch ; <- century BCD
0000A803 240F                 <1>       and    al, 0Fh
0000A805 88EC                 <1>       mov    ah, ch
0000A807 C0EC04               <1>       shr    ah, 4
0000A80A D50A                 <1>       aad
0000A80C 88C5                 <1>       mov    ch, al ; -> century
                              <1>
0000A80E 88C8                 <1>       mov    al, cl ; <- year BCD
0000A810 240F                 <1>       and    al, 0Fh
0000A812 88CC                 <1>       mov    ah, cl
0000A814 C0EC04               <1>       shr    ah, 4
0000A817 D50A                 <1>       aad
0000A819 88C1                 <1>       mov    cl, al ; -> year
                              <1>
0000A81B 88E8                 <1>       mov    al, ch
0000A81D B464                 <1>       mov    ah, 100
0000A81F F6E4                 <1>       mul    ah
0000A821 30ED                 <1>       xor    ch, ch
0000A823 6601C8               <1>       add    ax, cx
0000A826 662DBC07             <1>       sub    ax, 1980 ; ms-dos epoch
0000A82A 6689C1               <1>       mov    cx, ax
                              <1>
0000A82D 88F0                 <1>       mov    al, dh ; <- month in bcd
0000A82F 240F                 <1>       and    al, 0Fh
0000A831 88F4                 <1>       mov    ah, dh
0000A833 C0EC04               <1>       shr    ah, 4
0000A836 D50A                 <1>       aad
0000A838 88C6                 <1>       mov    dh, al ; -> month
                              <1>
0000A83A 88D0                 <1>       mov    al, dl ; <- day BCD
0000A83C 240F                 <1>       and    al, 0Fh
0000A83E 88D4                 <1>       mov    ah, dl
0000A840 C0EC04               <1>       shr    ah, 4
0000A843 D50A                 <1>       aad
0000A845 88C2                 <1>       mov    dl, al ; -> day
                              <1>
0000A847 88C8                 <1>       mov    al, cl ; count of years from 1980
0000A849 66C1E004             <1>       shl    ax, 4
0000A84D 08F0                 <1>       or     al, dh ; month of year, 1 to 12
0000A84F 66C1E005             <1>       shl    ax, 5
0000A853 08D0                 <1>       or     al, dl ; day of year, 1 to 31
                              <1>
0000A855 6650                 <1>       push   ax ; push date
                              <1>
0000A857 B402                 <1>       mov    ah, 02h ; Return Current Time
0000A859 E81CB1FFFF           <1>       call   int1Ah
                              <1>
0000A85E 88E8                 <1>       mov    al, ch ; <- hours BCD
0000A860 240F                 <1>       and    al, 0Fh
0000A862 88EC                 <1>       mov    ah, ch
0000A864 C0EC04               <1>       shr    ah, 4
0000A867 D50A                 <1>       aad
0000A869 88C5                 <1>       mov    ch, al ; -> hours
                              <1>
0000A86B 88C8                 <1>       mov    al, cl ; <- minutes BCD
0000A86D 240F                 <1>       and    al, 0Fh
0000A86F 88CC                 <1>       mov    ah, cl
0000A871 C0EC04               <1>       shr    ah, 4
0000A874 D50A                 <1>       aad
0000A876 88C1                 <1>       mov    cl, al ; -> minutes
                              <1>
0000A878 88F0                 <1>       mov    al, dh ; <- seconds BCD
0000A87A 240F                 <1>       and    al, 0Fh
0000A87C 88F4                 <1>       mov    ah, dh
0000A87E C0EC04               <1>       shr    ah, 4
0000A881 D50A                 <1>       aad
0000A883 88C6                 <1>       mov    dh, al ; -> seconds
                              <1>
0000A885 88E8                 <1>       mov    al, ch ; hours
0000A887 66C1E006             <1>       shl    ax, 6
0000A88B 08C8                 <1>       or     al, cl ; minutes
0000A88D 66C1E005             <1>       shl    ax, 5
0000A891 D0EE                 <1>       shr    dh, 1 ; 2 seconds
                              <1>       ; There is a bug in TRDOS v1 here !
                              <1>       ; it was 'or al, dl' !
0000A893 08F0                 <1>       or     al, dh ; seconds
                              <1>
0000A895 665A                 <1>       pop    dx ; pop date
                              <1>
0000A897 C3                   <1>       retn
                              <1>
                              <1> save_directory_buffer:
                              <1>       ; 15/10/2016
                              <1>       ; 23/03/2016
                              <1>       ; 26/02/2016
                              <1>       ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)
                              <1>       ; 01/08/2011
                              <1>       ; 14/03/2010
                              <1>       ; INPUT ->
                              <1>       ;      none
                              <1>       ; OUTPUT ->
                              <1>       ;  cf = 0 -> write OK...
                              <1>       ;  cf = 1 -> error code in AL (EAX)
                              <1>       ;  cf = 1 & AL = 0Dh => CH & CL = FS & FAT type
                              <1>       ;  EBX = Directory Buffer Address
```

```
                             <1>       ;
                             <1>       ;  (EAX, ECX, EDX will be modified)
                             <1>
0000A898 BB00000800          <1>       mov    ebx, Directory_Buffer
0000A89D 803D[28610100]02    <1>       cmp    byte [DirBuff_ValidData], 2
0000A8A4 7403                <1>       je     short loc_save_dir_buffer
0000A8A6 31C0                <1>       xor    eax, eax
0000A8A8 C3                  <1>       retn
                             <1>
                             <1> loc_save_dir_buffer:
0000A8A9 56                  <1>       push   esi
0000A8AA 31DB                <1>       xor    ebx, ebx
0000A8AC 8A3D[26610100]      <1>         mov     bh, [DirBuff_DRV]
0000A8B2 80EF41              <1>       sub    bh, 'A'
0000A8B5 BE00010900          <1>         mov     esi, Logical_DOSDisks
0000A8BA 01DE                <1>       add    esi, ebx
0000A8BC 668B4E03            <1>         mov     cx, [esi+LD_FATType]
                             <1>       ; CH = FS Type (A1h for FS)
                             <1>       ; CL = FAT Type (0 for FS)
0000A8C0 08C9                <1>       or     cl, cl
0000A8C2 7433                <1>       jz     short loc_save_dir_buff_stc_retn
                             <1>
                             <1> loc_save_dir_buffer_check_cluster_no:
0000A8C4 A1[2D610100]        <1>       mov    eax, [DirBuff_Cluster]
0000A8C9 28FF                <1>       sub    bh, bh ; ebx = 0
0000A8CB 09C0                <1>       or     eax, eax
0000A8CD 7540                <1>       jnz    short loc_save_sub_dir_buffer
0000A8CF 8A25[27610100]      <1>       mov    ah, [DirBuff_FATType]
0000A8D5 FEC3                <1>       inc    bl ;  bl = 1
0000A8D7 38DC                <1>       cmp    ah, bl
0000A8D9 721D                <1>       jb     short loc_save_dir_buff_inv_data_retn
0000A8DB FEC3                <1>       inc    bl ; bl = 2
0000A8DD 38E3                <1>       cmp    bl, ah
0000A8DF 7217                <1>       jb     short loc_save_dir_buff_inv_data_retn
                             <1>
                             <1> loc_save_root_dir_buffer:
0000A8E1 668B5E17            <1>       mov    bx, [esi+LD_BPB+RootDirEnts]
0000A8E5 6683C30F            <1>       add    bx, 15
0000A8E9 66C1EB04            <1>       shr    bx, 4 ; 16 dir entries per sector
0000A8ED 6609DB              <1>       or     bx, bx
0000A8F0 7405                <1>       jz     short loc_save_dir_buff_stc_retn
                             <1>       ;mov   ecx, ebx
0000A8F2 8B4664              <1>       mov    eax, [esi+LD_ROOTBegin] ; 26/02/2016
0000A8F5 EB23                <1>       jmp    short loc_write_directory_to_disk
                             <1>
                             <1> loc_save_dir_buff_stc_retn:
0000A8F7 F9                  <1>       stc
                             <1> loc_save_dir_buff_inv_data_retn:
                             <1>       ; 15/10/2016 (0Dh -> 29)
0000A8F8 B01D                <1>       mov    al, 29 ; Invalid data !
0000A8FA C605[28610100]00    <1>       mov    byte [DirBuff_ValidData], 0
0000A901 EB05                <1>       jmp    short loc_save_dir_buff_retn
                             <1>
                             <1> loc_write_directory_to_disk_err:
                             <1>       ; 15/10/2016 (disk write error code, 1Dh -> 18)
0000A903 B812000000          <1>       mov    eax, 18 ; Drive not ready or write error
                             <1>
                             <1> loc_save_dir_buff_retn:
0000A908 BB00000800          <1>       mov    ebx, Directory_Buffer
0000A90D 5E                  <1>       pop    esi
0000A90E C3                  <1>       retn
                             <1>
                             <1> loc_save_sub_dir_buffer:
                             <1>       ; ebx  = 0
0000A90F 83E802              <1>       sub    eax, 2
0000A912 8A5E13              <1>       mov    bl, [esi+LD_BPB+SecPerClust]
0000A915 F7E3                <1>       mul    ebx
0000A917 034668              <1>         add     eax, [esi+LD_DATABegin]
                             <1>       ;mov   ecx, ebx
                             <1>
                             <1> loc_write_directory_to_disk:
0000A91A 89D9                <1>       mov    ecx, ebx
0000A91C BB00000800          <1>       mov    ebx, Directory_Buffer
0000A921 E8A34E0000          <1>       call   disk_write
0000A926 72DB                <1>       jc     short loc_write_directory_to_disk_err
                             <1>
                             <1> loc_save_dir_buff_validate_retn:
0000A928 C605[28610100]01    <1>       mov    byte [DirBuff_ValidData], 1
0000A92F 31C0                <1>       xor    eax, eax
                             <1>       ; 26/02/2016
0000A931 EBD5                <1>       jmp    short loc_save_dir_buff_retn
                             <1>
                             <1> update_parent_dir_lmdt:
                             <1>       ; 29/12/2017
                             <1>       ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 01/08/2011
                             <1>       ; 16/10/2010
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      none
                             <1>       ; OUTPUT ->
                             <1>       ;      (last modification date & time of the parent dir
                             <1>       ;      will be changed/updated)
                             <1>       ;
                             <1>       ; (EAX, EBX, ECX, EDX, EDI will be changed)
                             <1>
0000A933 29C0                <1>       sub    eax, eax
0000A935 8A25[FC580100]      <1>       mov    ah, [Current_Dir_Level]
0000A93B A0[FD580100]        <1>       mov    al, [Current_FATType]
0000A940 3C01                <1>       cmp    al, 1
0000A942 723A                <1>       jb     short loc_UPDLMDT_proc_retn
                             <1>
                             <1> loc_update_parent_dir_lm_date_time:
```

```
0000A944 08E4                <1>       or     ah, ah
0000A946 7436                <1>       jz     short loc_UPDLMDT_proc_retn
                             <1> 
0000A948 56                  <1>       push   esi ; *
0000A949 8825[A0630100]      <1>       mov    [UPDLMDT_CDirLevel], ah
0000A94F 8B15[F8580100]      <1>       mov    edx, [Current_Dir_FCluster]
0000A955 8915[A1630100]      <1>       mov    [UPDLMDT_CDirFCluster], edx
                             <1> 
0000A95B FECC                <1>       dec    ah
0000A95D B90C000000          <1>       mov    ecx, 12
0000A962 BE[5F610100]        <1>         mov     esi, PATH_Array
                             <1> 
0000A967 8825[FC580100]      <1>       mov    [Current_Dir_Level], ah
0000A96D 08E4                <1>       or     ah, ah
0000A96F 750E                <1>       jnz    short loc_update_parent_dir_lmdt_load_sub_dir_1
0000A971 803D[FD580100]02    <1>       cmp    byte [Current_FATType], 2
0000A978 770B                <1>       ja     short loc_update_parent_dir_lmdt_load_sub_dir_2
0000A97A 28C0                <1>       sub    al, al ; eax = 0
0000A97C EB0A                <1>       jmp    short loc_update_parent_dir_lmdt_load_sub_dir_3
                             <1> 
                             <1> loc_UPDLMDT_proc_retn:
0000A97E C3                  <1>       retn
                             <1> 
                             <1> loc_update_parent_dir_lmdt_load_sub_dir_1:
0000A97F B010                <1>       mov    al, 16
0000A981 F6E4                <1>       mul    ah
0000A983 01C6                <1>       add    esi, eax
                             <1> 
                             <1> loc_update_parent_dir_lmdt_load_sub_dir_2:
0000A985 8B460C              <1>       mov    eax, [esi+12] ; Parent Dir First Cluster
                             <1> 
                             <1> loc_update_parent_dir_lmdt_load_sub_dir_3:
0000A988 A3[F8580100]        <1>       mov    [Current_Dir_FCluster], eax
                             <1> 
0000A98D 83C610              <1>       add    esi, 16
0000A990 66BF[8662]          <1>       mov    di, Dir_File_Name
0000A994 F3A4                <1>       rep    movsb
                             <1> 
0000A996 BE00010900          <1>       mov    esi, Logical_DOSDisks
0000A99B 29DB                <1>       sub    ebx, ebx
0000A99D 8A3D[FE580100]      <1>       mov    bh, [Current_Drv]
0000A9A3 01DE                <1>       add    esi, ebx
0000A9A5 E88FF7FFFF          <1>       call   reload_current_directory
0000A9AA 7230                <1>       jc     short loc_update_parent_dir_lmdt_restore_cdirlevel
                             <1> 
                             <1> loc_update_parent_dir_lmdt_locate_dir:
0000A9AC BE[86620100]        <1>       mov    esi, Dir_File_Name
0000A9B1 6631C9              <1>       xor    cx, cx
0000A9B4 66B81008            <1>       mov    ax, 0810h ; Only directories
0000A9B8 E8B5F6FFFF          <1>         call    locate_current_dir_file
                             <1>       ; EDI = DirBuff Directory Entry Address
0000A9BD 721D                <1>       jc     short loc_update_parent_dir_lmdt_restore_cdirlevel
                             <1> 
0000A9BF E836FEFFFF          <1>       call   convert_current_date_time
0000A9C4 66895712            <1>       mov    [edi+18], dx ; Last Access Date
0000A9C8 66895718            <1>       mov    [edi+24], dx ; Last Write Date
0000A9CC 66894716            <1>       mov    [edi+22], ax ; Last Write Time
                             <1> 
0000A9D0 C605[28610100]02    <1>       mov    byte [DirBuff_ValidData], 2
0000A9D7 E8BCFEFFFF          <1>       call   save_directory_buffer
                             <1>       ; 29/12/2017
                             <1>       ;jc    short loc_update_parent_dir_lmdt_restore_cdirlevel
                             <1>       ;xor   al, al
                             <1> loc_update_parent_dir_lmdt_restore_cdirlevel:
                             <1>       ;current directory level restoration
0000A9DC 8A25[A0630100]      <1>       mov    ah, [UPDLMDT_CDirLevel]
0000A9E2 8825[FC580100]      <1>       mov    [Current_Dir_Level], ah
0000A9E8 8B15[A1630100]      <1>         mov     edx, [UPDLMDT_CDirFCluster]
0000A9EE 8915[F8580100]      <1>       mov    [Current_Dir_FCluster], edx
                             <1> 
0000A9F4 5E                  <1>       pop    esi ; *
0000A9F5 C3                  <1>       retn
                             <1> 
                             <1> delete_longname:
                             <1>       ; 27/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 01/08/2011 (DIR.ASM, 'proc_delete_longname')
                             <1>       ; 14/03/2010
                             <1>       ; INPUT ->
                             <1>       ;      EAX = Directory Entry (Index) Number (< 65536)
                             <1>       ; OUTPUT ->
                             <1>       ;      cf = 0 -> OK  (EAX = 0)
                             <1>       ;      cf = 1 -> error code in EAX (AL)
                             <1>       ;
                             <1>       ; (Modified registers: EAX, EDX, ECX, EBX, EDI)
                             <1> 
0000A9F6 66A3[D0630100]      <1>       mov    [DLN_EntryNumber], ax
0000A9FC C605[D2630100]40    <1>         mov     byte [DLN_40h], 40h
                             <1> 
0000AA03 E858000000          <1>       call   locate_current_dir_entry
0000AA08 7308                <1>       jnc    short loc_dln_check_attributes
0000AA0A C3                  <1>       retn
                             <1> 
                             <1> loc_dln_longname_not_found:
0000AA0B B802000000          <1>       mov    eax, 2
0000AA10 F9                  <1>       stc
0000AA11 C3                  <1>       retn
                             <1> 
                             <1> loc_dln_check_attributes:
0000AA12 B00F                <1>       mov    al, 0Fh  ; long name
0000AA14 8A670B              <1>       mov    ah, [edi+0Bh] ; dir entry attributes
0000AA17 38C4                <1>       cmp    ah, al
0000AA19 75F0                <1>       jne    short loc_dln_longname_not_found
0000AA1B 8A27                <1>       mov    ah, [edi]
```

```
0000AA1D 2A25[D2630100]      <1>        sub    ah, [DLN_40h]
0000AA23 76E6                <1>        jna    short loc_dln_longname_not_found
0000AA25 80FC14              <1>        cmp    ah, 14h ; 84-64=20 -> 20*13=260 bytes
0000AA28 77E1                <1>        ja     short loc_dln_longname_not_found
                             <1> 
0000AA2A C607E5              <1>        mov    byte [edi], 0E5h  ; deleted sign
0000AA2D C605[28610100]02    <1>        mov    byte [DirBuff_ValidData], 2 ; changed/write sign
0000AA34 C605[D2630100]00    <1>        mov    byte [DLN_40h], 0 ; 40h -> 0
                             <1> 
                             <1> loc_dln_delete_next_ln_entry:
0000AA3B 80FC01              <1>        cmp    ah, 1
0000AA3E 7616                <1>        jna    short loc_dln_longname_retn
                             <1> loc_dln_delete_next_ln_entry_0:
0000AA40 66FF05[D0630100]    <1>        inc    word [DLN_EntryNumber]
0000AA47 0FB705[D0630100]    <1>        movzx  eax, word [DLN_EntryNumber]
0000AA4E E80D000000          <1>        call   locate_current_dir_entry
0000AA53 73BD                <1>        jnc    short loc_dln_check_attributes
                             <1> 
                             <1> loc_dln_longname_stc_retn:
0000AA55 C3                  <1>        retn
                             <1> 
                             <1> loc_dln_longname_retn:
                             <1>        ;cmp   byte [DirBuff_ValidData], 2
                             <1>        ;jne   short loc_dln_longname_retn_xor_eax
0000AA56 E83DFEFFFF          <1>        call   save_directory_buffer
0000AA5B 72F8                <1>        jc     short loc_dln_longname_stc_retn
                             <1> 
                             <1> loc_dln_longname_retn_xor_eax:
0000AA5D 31C0                <1>        xor    eax, eax
0000AA5F C3                  <1>        retn
                             <1> 
                             <1> locate_current_dir_entry:
                             <1>        ; 16/10/2016
                             <1>        ; 15/10/2016
                             <1>        ; 23/03/2016
                             <1>        ; 27/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>        ; 01/08/2011 (DIR.ASM, 'proc_locate_current_dir_entry')
                             <1>        ; 07/03/2010
                             <1>        ; INPUT ->
                             <1>        ;      EAX = Directory Entry (Index) Number (< 65536)
                             <1>        ; OUTPUT ->
                             <1>        ;      EDI = Directory Entry Address
                             <1>        ;      EAX = Cluster Number of Directory Buffer
                             <1>        ;      EBX = Directory Buffer Entry Offset
                             <1>        ;      ECX = DirBuff Valid Data identifier (CL)
                             <1>        ;      If CF = 0 and CL = 2 then
                             <1>        ;         directory buffer modified and
                             <1>        ;         must be written to disk.
                             <1>        ;      If CF = 0  and CL = 1 then
                             <1>        ;         dir buffer has been written to disk, already.
                             <1>        ;      CF = 1 -> Error code in EAX (AL)
                             <1>        ;
                             <1>        ; (Modified registers: EAX, EDX, ECX, EBX, EDI)
                             <1> 
                             <1> loc_locate_current_dir_entry:
0000AA60 56                  <1>        push   esi
0000AA61 89C1                <1>        mov    ecx, eax
0000AA63 BA20000000          <1>        mov    edx, 32
0000AA68 F7E2                <1>        mul    edx
0000AA6A A3[DC630100]        <1>        mov    [LCDE_ByteOffset], eax
0000AA6F 31DB                <1>        xor    ebx, ebx
0000AA71 8A3D[FE580100]      <1>        mov    bh, [Current_Drv]
0000AA77 A0[26610100]        <1>          mov     al, [DirBuff_DRV]
0000AA7C 2C41                <1>        sub    al, 'A'
0000AA7E BE00010900          <1>          mov     esi, Logical_DOSDisks
0000AA83 01DE                <1>        add    esi, ebx
0000AA85 38C7                <1>        cmp    bh, al
0000AA87 0F8592000000        <1>          jne     loc_lcde_reload_current_directory
                             <1> loc_lcde_cdl_check:
0000AA8D 803D[FC580100]00    <1>        cmp    byte [Current_Dir_Level], 0
0000AA94 772A                <1>        ja     short loc_lcde_calc_dirbuff_cluster_offset
                             <1>        ; 27/02/2016
                             <1>        ; TRDOS v1 has bug here for FAT32 fs !
                             <1>        ; (Root Directory Entries for FAT32 = 0)
0000AA96 807E0303            <1>        cmp    byte [esi+LD_FATType], 3   ; FAT32
0000AA9A 7324                <1>        jnb    short loc_lcde_calc_dirbuff_cluster_offset
                             <1> 
                             <1> loc_lcde_cdl_check_FAT12_16:
0000AA9C 668B4617            <1>        mov    ax, [esi+LD_BPB+RootDirEnts]
0000AAA0 6648                <1>        dec    ax
                             <1>        ;xor   dx, dx
0000AAA2 6639C8              <1>        cmp    ax, cx ; cx = Directory Entry (Index) Number
0000AAA5 720E                <1>        jb     short loc_lcde_stc_12h_retn
0000AAA7 66890D[D4630100]    <1>        mov    [LCDE_EntryIndex], cx
0000AAAE 31C0                <1>        xor    eax, eax
0000AAB0 E993000000          <1>          jmp     loc_lcde_check_dir_buffer_cluster
                             <1> 
                             <1> loc_lcde_stc_12h_retn:
0000AAB5 5E                  <1>        pop    esi
0000AAB6 89CB                <1>        mov    ebx, ecx
0000AAB8 89D1                <1>        mov    ecx, edx
                             <1>        ; 16/10/2016 (12h -> 12)
0000AABA B80C000000          <1>        mov    eax, 12 ; No more files
0000AABF C3                  <1>        retn
                             <1> 
                             <1> loc_lcde_calc_dirbuff_cluster_offset:
0000AAC0 8A5E13              <1>        mov    bl, [esi+LD_BPB+SecPerClust]
0000AAC3 30FF                <1>        xor    bh, bh
0000AAC5 668B4611            <1>        mov    ax, [esi+LD_BPB+BytesPerSec]
0000AAC9 66F7E3              <1>        mul    bx
0000AACC 6609D2              <1>        or     dx, dx ; If bytes per cluster > 32KB it is invalid
0000AACF 755D                <1>        jnz    short loc_lcde_invalid_format
                             <1>        ;mov   ecx, eax
```

```
0000AAD1 6689C1              <1>         mov    cx, ax ; BYTES PER CLUSTER
0000AAD4 A1[DC630100]        <1>         mov    eax, [LCDE_ByteOffset]
                             <1>         ;sub   edx, edx
0000AAD9 F7F1                <1>         div    ecx
0000AADB 3DFFFF0000          <1>         cmp    eax, 65535
0000AAE0 774C                <1>         ja     short loc_lcde_invalid_format
                             <1> 
                             <1>         ; cluster sequence number of directory (< 65536)
0000AAE2 66A3[D6630100]      <1>         mov    [LCDE_ClusterSN], ax
                             <1> 
0000AAE8 6689D0              <1>         mov    ax, dx ; byte offset in cluster (directory buffer)
0000AAEB 66BB2000            <1>         mov    bx, 32 ; ; 1 dir entry = 32 bytes
0000AAEF 6629D2              <1>           sub     dx, dx  ; 0
0000AAF2 66F7F3              <1>         div    bx
0000AAF5 66A3[D4630100]      <1>         mov    [LCDE_EntryIndex], ax ; dir entry index/sequence number
                             <1>                                ; (in directory buffer/cluster)
                             <1> loc_lcde_get_current_sub_dir_fcluster:
0000AAFB A1[F8580100]        <1>         mov    eax, [Current_Dir_FCluster]
                             <1> 
                             <1> loc_lcde_get_next_cluster:
0000AB00 66833D[D6630100]00  <1>         cmp    word [LCDE_ClusterSN], 0
0000AB08 763E                <1>         jna    short loc_lcde_check_dir_buffer_cluster
0000AB0A A3[D8630100]        <1>         mov    [LCDE_Cluster], eax
0000AB0F E834100000          <1>         call   get_next_cluster
0000AB14 7220                <1>         jc     short loc_lcde_check_gnc_error
0000AB16 66FF0D[D6630100]    <1>         dec    word [LCDE_ClusterSN]
0000AB1D EBE1                <1>         jmp    short loc_lcde_get_next_cluster
                             <1> 
                             <1> loc_lcde_reload_current_directory:
0000AB1F 51                  <1>         push   ecx
0000AB20 E814F6FFFF          <1>         call   reload_current_directory
0000AB25 59                  <1>         pop    ecx
0000AB26 0F8361FFFFFF        <1>           jnc      loc_lcde_cdl_check
0000AB2C 5E                  <1>         pop    esi
0000AB2D C3                  <1>         retn
                             <1> 
                             <1> loc_lcde_invalid_format:
                             <1>         ; 15/10/2016 (0Bh -> 28)
0000AB2E B81C000000          <1>         mov    eax, 28 ; Invalid Format !
                             <1> loc_lcde_drive_not_ready_read_err:
0000AB33 F9                  <1>         stc
0000AB34 5E                  <1>         pop    esi
0000AB35 C3                  <1>         retn
                             <1> 
                             <1> loc_lcde_check_gnc_error:
0000AB36 09C0                <1>         or     eax, eax
0000AB38 75F9                <1>         jnz    short loc_lcde_drive_not_ready_read_err
0000AB3A 66FF0D[D6630100]    <1>         dec    word [LCDE_ClusterSN]
0000AB41 75EB                <1>         jnz    short loc_lcde_invalid_format
0000AB43 A1[D8630100]        <1>         mov    eax, [LCDE_Cluster]
                             <1> 
                             <1> loc_lcde_check_dir_buffer_cluster:
0000AB48 3B05[2D610100]      <1>         cmp    eax, [DirBuff_Cluster]
0000AB4E 755C                <1>         jne    short loc_lcde_load_dir_cluster
0000AB50 803D[28610100]00    <1>         cmp    byte [DirBuff_ValidData], 0
0000AB57 7727                <1>         ja     short lcde_check_dir_buffer_cluster_next
0000AB59 803D[FC580100]00    <1>         cmp    byte [Current_Dir_Level], 0
0000AB60 775F                <1>         ja     short loc_lcde_load_dir_cluster_0
                             <1>         ; 27/02/2016
                             <1>         ; TRDOS v1 has bug here for FAT32 fs !
0000AB62 807E0303            <1>         cmp    byte [esi+LD_FATType], 3  ; FAT32
0000AB66 7359                <1>         jnb    short loc_lcde_load_dir_cluster_0
                             <1>         ;
0000AB68 0FB74E17            <1>         movzx  ecx, word [esi+LD_BPB+RootDirEnts]
0000AB6C 6683C10F            <1>         add    cx, 15 ; round up (16 entries per sector)
0000AB70 66C1E904            <1>         shr    cx, 4 ; 1 sector contains 16 dir entries
                             <1> 
0000AB74 8B4664              <1>           mov     eax, [esi+LD_ROOTBegin]
0000AB77 EB54                <1>         jmp    short loc_lcde_load_dir_cluster_1
                             <1> 
                             <1> loc_lcde_validate_dirBuff:
0000AB79 C605[28610100]01    <1>         mov    byte [DirBuff_ValidData], 1
                             <1> 
                             <1> lcde_check_dir_buffer_cluster_next:
0000AB80 0FB71D[D4630100]    <1>         movzx  ebx, word [LCDE_EntryIndex]
0000AB87 663B1D[2B610100]    <1>         cmp    bx, [DirBuff_LastEntry]
0000AB8E 779E                <1>         ja     short loc_lcde_invalid_format
0000AB90 B820000000          <1>         mov    eax, 32
0000AB95 F7E3                <1>         mul    ebx
                             <1>         ;or    edx, edx
                             <1>         ;jnz   short loc_lcde_invalid_format
                             <1> 
0000AB97 BF00000800          <1>         mov    edi, Directory_Buffer
0000AB9C 01C7                <1>         add    edi, eax ; add entry offset to buffer address
                             <1> 
                             <1> loc_lcde_dir_buffer_last_check:
0000AB9E A1[2D610100]        <1>         mov    eax, [DirBuff_Cluster]
0000ABA3 0FB60D[28610100]    <1>         movzx  ecx, byte [DirBuff_ValidData]
                             <1> 
                             <1> loc_lcde_retn:
0000ABAA 5E                  <1>         pop    esi
0000ABAB C3                  <1>         retn
                             <1> 
                             <1> loc_lcde_load_dir_cluster:
                             <1>         ;cmp   byte [DirBuff_ValidData], 2
                             <1>         ;jne   short loc_lcde_load_dir_cluster_n2
0000ABAC 50                  <1>         push   eax
0000ABAD E8E6FCFFFF          <1>         call   save_directory_buffer
0000ABB2 58                  <1>         pop    eax
0000ABB3 72F5                <1>         jc     short loc_lcde_retn
                             <1> 
                             <1> loc_lcde_load_dir_cluster_n2:
0000ABB5 C605[28610100]00    <1>         mov    byte [DirBuff_ValidData], 0
```

```
0000ABBC A3[2D610100]        <1>       mov    [DirBuff_Cluster], eax
                             <1>
                             <1> loc_lcde_load_dir_cluster_0:
0000ABC1 83E802              <1>       sub    eax, 2
0000ABC4 0FB64E13            <1>       movzx  ecx, byte [esi+LD_BPB+SecPerClust]
0000ABC8 F7E1                <1>       mul    ecx
0000ABCA 034668              <1>         add     eax, [esi+LD_DATABegin]
                             <1>
                             <1> loc_lcde_load_dir_cluster_1:
0000ABCD BB00000800          <1>       mov    ebx, Directory_Buffer
                             <1>       ; ecx = sector count
0000ABD2 E8014C0000          <1>       call   disk_read
0000ABD7 73A0                <1>       jnc    short loc_lcde_validate_dirBuff
                             <1>
                             <1>       ; 15/10/2016
                             <1>       ; (Disk read error instead of drv not ready err)
0000ABD9 B811000000          <1>       mov    eax, 17 ; Drive not ready or read error !
0000ABDE EBCA                <1>       jmp    short loc_lcde_retn
                             <1>
                             <1>
                             <1> remove_file:
                             <1>       ; 15/10/2016
                             <1>       ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 10/04/2011 (FILE.ASM, 'proc_delete_file')
                             <1>       ; 09/08/2010
                             <1>       ; INPUT ->
                             <1>       ;     EDI = Directory Buffer Entry Address
                             <1>       ;      CX = Directory Buffer Entry Counter/Index
                             <1>       ;      BL = Longname Entry Length
                             <1>       ;      BH = Logical DOS Drive Number
                             <1>
0000ABE0 29C0                <1>       sub    eax, eax
0000ABE2 88FC                <1>       mov    ah, bh
0000ABE4 BE00010900          <1>       mov    esi, Logical_DOSDisks
0000ABE9 01C6                <1>       add    esi, eax
                             <1>
0000ABEB 807E0301            <1>       cmp    byte [esi+LD_FATType], 1
0000ABEF 7312                <1>       jnb    short loc_del_fat_file
                             <1>
0000ABF1 807E04A1            <1>       cmp    byte [esi+LD_FSType], 0A1h
0000ABF5 7406                <1>       je     short loc_del_fs_file
                             <1>
                             <1> loc_del_file_invalid_format:
0000ABF7 30E4                <1>       xor    ah, ah
                             <1>       ; 15/10/2016 (0Bh -> 28)
0000ABF9 B01C                <1>       mov    al, 28  ; Invalid Format
0000ABFB F9                  <1>       stc
0000ABFC C3                  <1>       retn
                             <1>
                             <1> loc_del_fs_file:
0000ABFD E83F0F0000          <1>       call   delete_fs_file
0000AC02 C3                  <1>       retn
                             <1>
                             <1> loc_del_fat_file:
0000AC03 E808000000          <1>       call   delete_directory_entry
0000AC08 7205                <1>       jc     short loc_del_file_err_retn
                             <1>
                             <1> loc_delfile_unlink_cluster_chain:
0000AC0A E863170000          <1>       call   truncate_cluster_chain
                             <1>       ;jc    short loc_del_file_err_retn
                             <1>
                             <1> loc_delfile_return:
                             <1> loc_del_file_err_retn:
0000AC0F C3                  <1>       retn
                             <1>
                             <1> delete_directory_entry:
                             <1>       ; 15/10/2016
                             <1>       ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 01/08/2011 (DIR.ASM, 'proc_delete_directory_entry')
                             <1>       ; 10/04/2011
                             <1>       ; INPUT ->
                             <1>       ;     ESI = Logical Dos Drive Descripton Table Address
                             <1>       ;     EDI = Directory Buffer Entry Address
                             <1>       ;      CX = Directory Buffer Entry Counter/Index
                             <1>       ;      BL = Longname Entry Length
                             <1>       ; OUTPUT ->
                             <1>       ;     ESI = Logical dos drive descripton table address
                             <1>       ;     EAX = First cluster to be truncated/unlinked
                             <1>       ;       CF = 1 -> Error code in EAX (AL)
                             <1>       ;       CF = 0 & BH <> 0 -> LMDT write error  (BH = 1)
                             <1>       ;       CF = 0 & BL <> 0 -> Long name delete error (BL = FFh)
                             <1>       ;
                             <1>       ;  (EDI, EBX, ECX register contents will be changed)
                             <1>
0000AC10 881D[6A630100]      <1>       mov    [DelFile_LNEL], bl
0000AC16 66890D[68630100]    <1>       mov    [DelFile_EntryCounter], cx
                             <1>
0000AC1D 668B4714            <1>       mov    ax, [edi+20] ; First Cluster High Word
0000AC21 C1E010              <1>       shl    eax, 16
0000AC24 668B471A            <1>       mov    ax, [edi+26] ; First Cluster Low Word
                             <1>
0000AC28 A3[64630100]        <1>       mov    [DelFile_FCluster], eax
                             <1>
                             <1> loc_del_short_name:
0000AC2D C607E5              <1>       mov    byte [edi], 0E5h  ; Deleted sign
                             <1>
0000AC30 C605[28610100]02    <1>       mov    byte [DirBuff_ValidData], 2
0000AC37 E85CFCFFFF          <1>       call   save_directory_buffer
0000AC3C 723D                <1>       jc     short loc_delete_direntry_err_return
                             <1>
                             <1> loc_del_long_name:
0000AC3E 0FB615[6A630100]    <1>       movzx  edx, byte [DelFile_LNEL]
0000AC45 08D2                <1>       or     dl, dl
```

```
0000AC47 7416                <1>       jz     short loc_del_dir_entry_update_parent_dir_lm_date
                             <1> 
0000AC49 8835[6A630100]      <1>       mov    byte [DelFile_LNEL], dh ; 0
                             <1> 
0000AC4F 0FB705[68630100]    <1>       movzx  eax,  word [DelFile_EntryCounter]
0000AC56 29D0                <1>       sub    eax, edx
                             <1>       ;jnc   short loc_del_long_name_continue
0000AC58 7205                <1>       jc     short loc_del_dir_entry_update_parent_dir_lm_date
                             <1> 
                             <1> ;loc_del_direntry_inv_data_return: ; 15/10/2016 (0Dh -> 29)
                             <1> ;     mov    eax, 29 ; 0Dh (TRDOS 8086) ; Invalid data
                             <1> ;     retn
                             <1> 
                             <1> loc_del_long_name_continue:
                             <1>       ; AX = Directory Entry Number of the long name last entry
0000AC5A E897FDFFFF          <1>       call   delete_longname
                             <1>       ;jc    short loc_delete_direntry_err_return
                             <1> 
                             <1> loc_del_dir_entry_update_parent_dir_lm_date:
0000AC5F 801D[6A630100]00    <1>       sbb    byte [DelFile_LNEL], 0 ; 0FFh if cf = 1
                             <1> 
0000AC66 E8C8FCFFFF          <1>       call   update_parent_dir_lmdt
0000AC6B B700                <1>       mov    bh, 0
0000AC6D 80D700              <1>       adc    bh, 0
                             <1> 
0000AC70 8A1D[6A630100]      <1>       mov    bl, byte [DelFile_LNEL]
                             <1> 
                             <1> loc_delete_direntry_return:
0000AC76 A1[64630100]        <1>       mov    eax, [DelFile_FCluster]
                             <1> loc_delete_direntry_err_return:
0000AC7B C3                  <1>       retn
                             <1> 
                             <1> rename_directory_entry:
                             <1>       ; 13/11/2017
                             <1>       ; 15/10/2016
                             <1>       ; 06/03/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 01/08/2011 (DIR.ASM, 'proc_rename_directory_entry')
                             <1>       ; 19/11/2010
                             <1>       ; INPUT -> (Current Directory)
                             <1>       ;      CX = Directory Entry Number
                             <1>       ;      EAX = First Cluster number of file or directory
                             <1>       ;      EBX = Longname Length (dir entry count) (< 256)
                             <1>       ;      ESI = New file (or directory) name (no path).
                             <1>       ;           (ASCIIZ string)
                             <1>       ; OUTPUT ->
                             <1>       ;      CF = 0 -> successfull
                             <1>       ;      CF = 1 -> error code in EAX (AL)
                             <1>       ;
                             <1>       ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
                             <1> 
0000AC7C 803D[FD580100]00    <1>       cmp    byte [Current_FATType], 0
0000AC83 7706                <1>       ja     short loc_rename_directory_entry
                             <1> 
0000AC85 E8B80E0000          <1>       call   rename_fs_file_or_directory
0000AC8A C3                  <1>       retn
                             <1> 
                             <1> loc_rename_directory_entry:
0000AC8B 881D[6A630100]      <1>       mov    [DelFile_LNEL], bl
0000AC91 66890D[68630100]    <1>       mov    [DelFile_EntryCounter], cx
0000AC98 A3[64630100]        <1>       mov    [DelFile_FCluster], eax
                             <1> 
0000AC9D 0FB7C1              <1>       movzx  eax, cx
0000ACA0 E8BBFDFFFF          <1>       call   locate_current_dir_entry
0000ACA5 7308                <1>       jnc    short loc_rename_direntry_check_fcluster
                             <1> 
                             <1> loc_rename_direntry_pop_retn:
0000ACA7 C3                  <1>       retn
                             <1> 
                             <1> loc_rename_direntry_pop_invd_retn:
0000ACA8 F9                  <1>       stc
                             <1> loc_rename_direntry_invd_retn:
                             <1>       ; 15/10/2016 (0Dh -> 29)
0000ACA9 B81D000000          <1>       mov    eax, 29 ; Invalid data
                             <1> loc_rename_retn:
0000ACAE C3                  <1>       retn
                             <1> 
                             <1> loc_rename_direntry_check_fcluster:
0000ACAF 668B5714            <1>       mov    dx, [edi+20] ; First Cluster HW
0000ACB3 C1E210              <1>       shl    edx, 16 ; 13/11/2017
0000ACB6 668B571A            <1>       mov    dx, [edi+26] ; First Cluster LW
0000ACBA 3B15[64630100]      <1>       cmp    edx, [DelFile_FCluster]
0000ACC0 75E6                <1>       jne    short loc_rename_direntry_pop_invd_retn
                             <1>       ; ESI = New file (or directory) name. (ASCIIZ string)
                             <1>       ; 06/03/2016
                             <1>       ; TRDOS v2 - NOTE: 'convert_file_name' procedure
                             <1>       ; has been modified for eliminating following situation.
                             <1>       ;
                             <1>       ; TRDOS v1 - NOTE: If file/dir name is more than 11 bytes
                             <1>       ; without a dot, attributes (edi+11) byte will be overwritten !
                             <1>       ; (Dot file name input must be proper for 11 byte dir entry
                             <1>       ;  type file name output.)
0000ACC2 E8A2F6FFFF          <1>       call   convert_file_name
                             <1> 
0000ACC7 C605[28610100]02    <1>         mov     byte [DirBuff_ValidData], 2
0000ACCE E8C5FBFFFF          <1>       call   save_directory_buffer
0000ACD3 72D9                <1>       jc     short loc_rename_retn
                             <1> 
                             <1> loc_rename_direntry_del_ln:
0000ACD5 0FB615[6A630100]    <1>       movzx  edx, byte [DelFile_LNEL]
0000ACDC 08D2                <1>       or     dl, dl
0000ACDE 7410                <1>       jz     short loc_rename_direntry_update_parent_dir_lm_date
                             <1> 
0000ACE0 0FB705[68630100]    <1>       movzx  eax, word [DelFile_EntryCounter]
```

```
0000ACE7 29D0                <1>       sub    eax, edx
0000ACE9 72BE                <1>       jc     short loc_rename_direntry_invd_retn
                             <1>
                             <1> loc_rename_direntry_del_ln_continue:
                             <1>       ; EAX = Directory Entry Number of the long name last entry
0000ACEB E806FDFFFF          <1>       call   delete_longname
                             <1>
                             <1> loc_rename_direntry_update_parent_dir_lm_date:
0000ACF0 E83EFCFFFF          <1>       call   update_parent_dir_lmdt
0000ACF5 31C0                <1>       xor    eax, eax
0000ACF7 C3                  <1>       retn
                             <1>
                             <1> move_source_file_to_destination_file:
                             <1>       ; 15/10/2016
                             <1>       ; 11/03/2016
                             <1>       ; 10/03/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 01/08/2011 (FILE.ASM)
                             <1>       ; 04/08/2010
                             <1>       ;
                             <1>       ;   Phase 1 -> Check destination file,
                             <1>       ;              'not found' is required
                             <1>       ;   Phase 2 -> Check source file
                             <1>       ;              'found' and proper attributes is required
                             <1>       ;   Phase 3 -> Make destination directory entry,
                             <1>       ;           add new dir cluster or section if it is required
                             <1>       ;   Phase 4 -> Delete source directory entry.
                             <1>       ;       cf = 1 causes to return before the phase 4.
                             <1>       ;    (source file protection against any possible errors)
                             <1>       ;
                             <1>       ; 08/05/2011 major modification
                             <1>       ;            -> destination file deleting is removed
                             <1>       ;            for msdos move/rename compatibility.
                             <1>       ;            (Access denied error will return if
                             <1>       ;            the destination file is found...)
                             <1>       ; INPUT ->
                             <1>       ;      ESI = Source File Pathname (Asciiz)
                             <1>       ;        EDI = Destination File Pathname (Asciiz)
                             <1>       ;        AL = 0 --> Interrupt (System call)
                             <1>       ;        AL > 0 --> Command Interpreter (Question)
                             <1>       ;        AL = 1 --> Question Phase
                             <1>       ;        AL = 2 --> Progress Phase
                             <1>       ; OUTPUT ->
                             <1>       ;      cf = 0 -> OK
                             <1>       ;        EAX = Destination directory first cluster
                             <1>       ;        ESI = Logical DOS drive description table
                             <1>       ;        EBX = Destination file structure offset
                             <1>       ;        CX = 0 (CX > 0 --> calculate free space error)
                             <1>       ;        cf = 1 -> Error code in EAX (AL)
                             <1>       ;
                             <1>       ;  (EDX, ECX, EBX, ESI, EDI will be changed)
                             <1>
0000ACF8 3C02                <1>       cmp    al, 2
0000ACFA 0F847F010000        <1>       je     msftdf_df2_check_directory
0000AD00 A2[EA640100]        <1>       mov    [move_cmd_phase], al
                             <1>
                             <1> msftdf_parse_sf_path:
                             <1>       ; ESI = ASCIIZ pathname (Source)
0000AD05 57                  <1>       push   edi
0000AD06 BF[E8630100]        <1>       mov    edi, SourceFile_Drv
0000AD0B E824F7FFFF          <1>       call   parse_path_name
0000AD10 5E                  <1>       pop    esi
0000AD11 7211                <1>       jc     short msftdf_psf_retn
                             <1>
                             <1> msftdf_parse_df_path:
                             <1>       ; ESI = ASCIIZ pathname   (Destination)
0000AD13 BF[68640100]        <1>       mov    edi, DestinationFile_Drv
0000AD18 E817F7FFFF          <1>       call   parse_path_name
0000AD1D 7306                <1>       jnc    short msftdf_check_sf_drv
                             <1>
0000AD1F 3C01                <1>       cmp    al, 1 ; File or directory name is not existing
0000AD21 7602                <1>       jna    short msftdf_check_sf_drv
                             <1>
                             <1> msftdf_stc_retn:
0000AD23 F9                  <1>       stc
                             <1> msftdf_psf_retn:
0000AD24 C3                  <1>       retn
                             <1>
                             <1> msftdf_check_sf_drv:
0000AD25 A0[E8630100]        <1>       mov    al, [SourceFile_Drv]
                             <1>
                             <1> msftdf_check_df_drv:
0000AD2A 8A15[68640100]      <1>       mov    dl, [DestinationFile_Drv]
                             <1>
                             <1> msftdf_compare_sf_df_drv:
0000AD30 29DB                <1>       sub    ebx, ebx
0000AD32 8A3D[FE580100]      <1>       mov    bh, [Current_Drv]
0000AD38 38C2                <1>       cmp    dl, al
0000AD3A 7409                <1>       je     short msftdf_check_sf_df_drv_ok
                             <1>
                             <1> msftdf_not_same_drv:
                             <1>         ; DL = source file's drive number
0000AD3C 88C6                <1>       mov    dh, al ; destination file's drive number
                             <1>       ; 15/10/2016 (11h -> 21)
0000AD3E B815000000          <1>       mov    eax, 21 ; Not the same drive
0000AD43 F9                  <1>       stc
0000AD44 C3                  <1>       retn
                             <1>
                             <1> msftdf_check_sf_df_drv_ok:
0000AD45 8815[EB640100]      <1>       mov    [msftdf_sf_df_drv], dl
                             <1>
0000AD4B 29C0                <1>         sub  eax, eax
0000AD4D 88D4                <1>       mov    ah, dl
0000AD4F 0500010900          <1>       add    eax, Logical_DOSDisks
```

```
0000AD54 A3[EC640100]         <1>       mov    [msftdf_drv_offset], eax
                              <1>
0000AD59 38FA                 <1>       cmp    dl, bh ; byte [Current_Drv]
0000AD5B 7407                 <1>       je     short msftdf_df_check_directory
                              <1>
                              <1> msftdf_change_drv:
0000AD5D E85EC1FFFF           <1>       call   change_current_drive
0000AD62 726D                 <1>       jc     short msftdf_df_error_retn
                              <1>
                              <1> msftdf_check_destination_file:
                              <1> msftdf_df_check_directory:
0000AD64 BE[69640100]         <1>       mov    esi, DestinationFile_Directory
0000AD69 803E20               <1>       cmp    byte [esi], 20h
0000AD6C 760F                 <1>       jna    short msftdf_df_find_1
                              <1>
                              <1> msftdf_df_change_directory:
0000AD6E FE05[D30C0100]       <1>       inc    byte [Restore_CDIR]
0000AD74 30E4                 <1>       xor    ah, ah ; CD_COMMAND sign -> 0
0000AD76 E8A3F0FFFF           <1>       call   change_current_directory
0000AD7B 7254                 <1>       jc     short msftdf_df_error_retn
                              <1>
                              <1> ;msftdf_df_change_prompt_dir_string:
                              <1> ;     call   change_prompt_dir_string
                              <1>
                              <1> msftdf_df_find_1:
0000AD7D BE[AA640100]         <1>         mov     esi, DestinationFile_Name
0000AD82 803E20               <1>       cmp    byte [esi], 20h
0000AD85 7631                 <1>       jna    short msftdf_df_copy_sf_name
                              <1>
                              <1> msftdf_df_find_2:
0000AD87 6631C0               <1>       xor    ax, ax ; DestinationFile_AttributesMask -> any/zero
0000AD8A E8D4D4FFFF           <1>       call   find_first_file
0000AD8F 0F838D000000         <1>       jnc    msftdf_permission_denied_retn
                              <1>
                              <1> msftdf_df_check_error_code:
                              <1>       ;cmp   eax, 2 ; File not found error
0000AD95 3C02                 <1>       cmp    al, 2
0000AD97 7537                 <1>       jne    short msftdf_df_stc_retn
                              <1>
                              <1> msftdf_df_check_fname:
                              <1>       ; 15/10/2016
0000AD99 BE[AA640100]         <1>       mov    esi, DestinationFile_Name ; *
0000AD9E E87ED8FFFF           <1>       call   check_filename
0000ADA3 7307                 <1>       jnc    short msftdf_convert_df_direntry_name
                              <1>       ; invalid file name chars !
0000ADA5 B81A000000           <1>       mov    eax, ERR_INV_FILE_NAME  ; 26
0000ADAA EB24                 <1>       jmp    short msftdf_df_stc_retn
                              <1>
                              <1> msftdf_convert_df_direntry_name:
                              <1>       ; mov  esi, DestinationFile_Name ; *
0000ADAC BF[BA640100]         <1>       mov    edi, DestinationFile_DirEntry
0000ADB1 E8B3F5FFFF           <1>       call   convert_file_name
0000ADB6 EB1A                 <1>       jmp    short msftdf_restore_current_dir_1
                              <1>
                              <1> msftdf_df_copy_sf_name:
0000ADB8 89F7                 <1>       mov    edi, esi
0000ADBA 57                   <1>       push   edi
0000ADBB BE[2A640100]         <1>         mov     esi, SourceFile_Name
0000ADC0 B90C000000           <1>       mov    ecx, 12
                              <1> msftdf_df_copy_sf_name_loop:
0000ADC5 AC                   <1>       lodsb
0000ADC6 AA                   <1>         stosb
0000ADC7 08C0                 <1>       or     al, al
0000ADC9 7402                 <1>       jz     short msftdf_df_copy_sf_name_ok
0000ADCB E2F8                 <1>         loop    msftdf_df_copy_sf_name_loop
                              <1> msftdf_df_copy_sf_name_ok:
0000ADCD 5E                   <1>       pop    esi
0000ADCE EBB7                 <1>       jmp    short msftdf_df_find_2
                              <1>
                              <1> msftdf_df_stc_retn:
0000ADD0 F9                   <1>       stc
                              <1> msftdf_restore_cdir_failed:
                              <1> msftdf_df_error_retn:
0000ADD1 C3                   <1>       retn
                              <1>
                              <1> msftdf_restore_current_dir_1:
0000ADD2 803D[D30C0100]00     <1>       cmp    byte [Restore_CDIR], 0
0000ADD9 760D                 <1>       jna    short msftdf_sf_check_directory
0000ADDB 8B35[EC640100]       <1>       mov    esi, [msftdf_drv_offset]
0000ADE1 E891C1FFFF           <1>       call   restore_current_directory
0000ADE6 72E9                 <1>       jc     short msftdf_restore_cdir_failed
                              <1>
                              <1> msftdf_sf_check_directory:
0000ADE8 BE[E9630100]         <1>       mov    esi, SourceFile_Directory
0000ADED 803E20               <1>       cmp    byte [esi], 20h
0000ADF0 760F                 <1>       jna    short msftdf_sf_find
                              <1> msftdf_sf_change_directory:
0000ADF2 FE05[D30C0100]       <1>       inc    byte [Restore_CDIR]
0000ADF8 30E4                 <1>       xor    ah, ah ; CD_COMMAND sign -> 0
0000ADFA E81FF0FFFF           <1>       call   change_current_directory
0000ADFF 7227                 <1>       jc     short msftdf_return
                              <1>
                              <1> ;msftdf_sf_change_prompt_dir_string:
                              <1> ;     call   change_prompt_dir_string
                              <1>
                              <1> msftdf_sf_find:
0000AE01 BE[2A640100]         <1>         mov     esi, SourceFile_Name  ; Offset 66
0000AE06 66B80018             <1>       mov    ax, 1800h ; Only files
0000AE0A E854D4FFFF           <1>       call   find_first_file
0000AE0F 7217                 <1>       jc     short msftdf_return
                              <1>
                              <1> msftdf_sf_ambgfn_check:
0000AE11 6609D2               <1>       or     dx, dx ; Ambiguous filename chars used sign (DX>0)
```

```
0000AE14 7407                <1>       jz     short msftdf_sf_found
                             <1> 
                             <1> msftdf_ambiguous_file_name_error:
0000AE16 B802000000          <1>       mov    eax, 2 ; File not found error
0000AE1B F9                  <1>       stc
0000AE1C C3                  <1>       retn
                             <1> 
                             <1> msftdf_sf_found:
0000AE1D 80E31F              <1>       and    bl, 1Fh ; Attributes, D-V-S-H-R
0000AE20 7416                <1>       jz     short msftdf_save_sf_structure
                             <1> 
                             <1> msftdf_permission_denied_retn:
0000AE22 B805000000          <1>       mov    eax, 05h ; Access (Permission) denied !
0000AE27 F9                  <1>       stc
                             <1> msftdf_rest_cdir_err_retn:
                             <1> msftdf_return:
0000AE28 C3                  <1>       retn
                             <1> 
                             <1> msftdf_phase_1_return:
0000AE29 31C0                <1>       xor    eax, eax
0000AE2B A2[EA640100]        <1>       mov    [move_cmd_phase], al ; 0
0000AE30 FEC0                <1>       inc    al ; mov al, 1
0000AE32 BB[7FAE0000]        <1>       mov    ebx, msftdf_df2_check_directory
                             <1>       ;mov   edx, 0FFFFFFFFh
0000AE37 C3                  <1>       retn
                             <1> 
                             <1> msftdf_save_sf_structure:
0000AE38 BE[F4620100]        <1>       mov    esi, FindFile_DirEntry
0000AE3D BF[3A640100]        <1>       mov    edi, SourceFile_DirEntry
0000AE42 B908000000          <1>       mov    ecx, 8
0000AE47 F3A5                <1>       rep    movsd
                             <1> 
                             <1> msftdf_df_copy_sf_parameters:
0000AE49 BE0B000000          <1>       mov    esi, 11
0000AE4E 89F7                <1>       mov    edi, esi
0000AE50 81C6[3A640100]      <1>       add    esi, SourceFile_DirEntry
0000AE56 81C7[BA640100]      <1>       add    edi, DestinationFile_DirEntry
                             <1>       ;mov   ecx, 21
0000AE5C B115                <1>       mov    cl, 21
0000AE5E F3A4                <1>       rep    movsb
                             <1> 
                             <1> msftdf_restore_current_dir_2:
0000AE60 803D[D30C0100]00    <1>       cmp    byte [Restore_CDIR], 0
0000AE67 760D                <1>       jna    short msftdf_df2_check_move_cmd_phase
0000AE69 8B35[EC640100]      <1>       mov    esi, [msftdf_drv_offset]
0000AE6F E803C1FFFF          <1>       call   restore_current_directory
0000AE74 72B2                <1>       jc     short msftdf_rest_cdir_err_retn
                             <1> 
                             <1> msftdf_df2_check_move_cmd_phase:
0000AE76 803D[EA640100]01    <1>       cmp    byte [move_cmd_phase], 1
0000AE7D 74AA                <1>       je     short msftdf_phase_1_return
                             <1> 
                             <1> msftdf_df2_check_directory:
0000AE7F BE[69640100]        <1>       mov    esi, DestinationFile_Directory
0000AE84 803E20              <1>       cmp    byte [esi], 20h
0000AE87 760F                <1>       jna    short msftdf_make_dfde_locate_ffe_on_directory
                             <1> msftdf_df2_change_directory:
0000AE89 FE05[D30C0100]      <1>       inc    byte [Restore_CDIR]
0000AE8F 30E4                <1>       xor    ah, ah ; CD_COMMAND sign -> 0
0000AE91 E888EFFFFF          <1>       call   change_current_directory
0000AE96 7290                <1>       jc     short msftdf_return
                             <1> 
                             <1> ;msftdf_df2_change_prompt_dir_string:
                             <1> ;     call   change_prompt_dir_string
                             <1> 
                             <1> msftdf_make_dfde_locate_ffe_on_directory:
                             <1>       ; Current directory fcluster <> Directory buffer cluster
                             <1>       ; Current directory will be reloaded by
                             <1>       ; 'locate_current_dir_file' procedure
                             <1>       ;
                             <1>       ;xor   ax, ax
0000AE98 31C0                <1>       xor    eax, eax
0000AE9A 89C1                <1>       mov    ecx, eax
0000AE9C 6649                <1>       dec    cx ; FFFFh
                             <1>              ; CX = FFFFh -> find first deleted or free entry
                             <1>              ; ESI would be ASCIIZ filename address if the call
                             <1>              ; would not be for first free or deleted dir entry
0000AE9E E8CFF1FFFF          <1>       call   locate_current_dir_file
0000AEA3 733F                <1>       jnc    msftdf_make_dfde_set_ff_dir_entry
                             <1> 
                             <1>       ;cmp   eax, 2
0000AEA5 3C02                <1>         cmp  al, 2
0000AEA7 7537                <1>       jne    short msftdf_error_retn
                             <1> 
                             <1> msftdf_add_new_dir_entry_check_fs:
0000AEA9 8B35[EC640100]      <1>       mov    esi, [msftdf_drv_offset]
0000AEAF A1[2D610100]        <1>       mov    eax, [DirBuff_Cluster]
0000AEB4 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
0000AEB8 7711                <1>       ja     short msftdf_add_new_subdir_cluster
                             <1> 
                             <1> msftdf_add_new_fs_subdir_section:
                             <1>       ;CL=0, CH=E5h --> deleted entry, CH=0 --> free entry
                             <1>         ;xor cx, cx
0000AEBA 30ED                <1>       xor    ch, ch ; cx = 0 --> add a new subdir section
0000AEBC E8830C0000          <1>       call   add_new_fs_section
0000AEC1 721E                <1>         jc   short msftdf_dsfde_error_retn
                             <1>       ;mov   [createfile_LastDirCluster], eax
                             <1> 
0000AEC3 E8A30E0000          <1>       call   load_FS_sub_directory
                             <1>       ;mov   ebx, Directory_Buffer
0000AEC8 7318                <1>       jnc    short msftdf_add_new_fs_subdir_section_ok
0000AECA C3                  <1>       retn
                             <1> 
```

```
                               <1> msftdf_add_new_subdir_cluster:
0000AECB E881150000            <1>       call   add_new_cluster
0000AED0 720F                  <1>       jc     short msftdf_dsfde_error_retn
                               <1> 
                               <1>       ;mov   [createfile_LastDirCluster], eax
                               <1> 
0000AED2 E8570E0000            <1>       call   load_FAT_sub_directory
0000AED7 7309                  <1>       jnc    short msftdf_add_new_subdir_cluster_ok
                               <1>       ; EBX = Directory buffer address
                               <1> 
                               <1> msftdf_ansdc_update_parent_dir_lmdt:
                               <1> msftdf_make_dfde_err_upd_pdir_lmdt:
0000AED9 50                    <1>       push   eax
0000AEDA E854FAFFFF            <1>       call   update_parent_dir_lmdt
0000AEDF 58                    <1>       pop    eax
                               <1> 
                               <1> msftdf_error_retn:
0000AEE0 F9                    <1>       stc
                               <1> msftdf_dsfde_restore_cdir_failed:
                               <1> msftdf_dsfde_error_retn:
0000AEE1 C3                    <1>       retn
                               <1> 
                               <1> msftdf_add_new_fs_subdir_section_ok:
                               <1> msftdf_add_new_subdir_cluster_ok:
0000AEE2 89DF                  <1>       mov    edi, ebx ; Directory buffer address
                               <1> 
                               <1> msftdf_make_dfde_set_ff_dir_entry:
0000AEE4 8B15[F8580100]        <1>       mov    edx, [Current_Dir_FCluster]
0000AEEA 8915[50650100]        <1>       mov    [createfile_FFCluster], edx
                               <1>       ; EDI = Directory entry offset
0000AEF0 BE[BA640100]          <1>       mov    esi, DestinationFile_DirEntry
0000AEF5 B908000000            <1>       mov    ecx, 8
0000AEFA F3A5                  <1>       rep    movsd
                               <1> 
0000AEFC C605[28610100]02      <1>       mov    byte [DirBuff_ValidData], 2
0000AF03 E890F9FFFF            <1>       call   save_directory_buffer
0000AF08 72CF                  <1>       jc     short msftdf_make_dfde_err_upd_pdir_lmdt
                               <1> 
                               <1> msftdf_make_dfde_update_pdir_lmdt:
0000AF0A E824FAFFFF            <1>       call   update_parent_dir_lmdt
                               <1> 
                               <1> msftdf_dsfde_restore_current_dir_1:
0000AF0F 803D[D30C0100]00      <1>       cmp    byte [Restore_CDIR], 0
0000AF16 760D                  <1>       jna    short msftdf_dsfde_check_directory
0000AF18 8B35[EC640100]        <1>       mov    esi, [msftdf_drv_offset]
0000AF1E E854C0FFFF            <1>       call   restore_current_directory
0000AF23 72BC                  <1>       jc     short msftdf_dsfde_restore_cdir_failed
                               <1> 
                               <1> msftdf_dsfde_check_directory:
0000AF25 BE[E9630100]          <1>       mov    esi, SourceFile_Directory
0000AF2A 803E20                <1>       cmp    byte [esi], 20h
0000AF2D 760F                  <1>       jna    short msftdf_dsfde_find_file
                               <1> 
                               <1> msftdf_dsfde_change_directory:
0000AF2F FE05[D30C0100]        <1>       inc    byte [Restore_CDIR]
0000AF35 28E4                  <1>       sub    ah, ah ; CD_COMMAND sign -> 0
0000AF37 E8E2EEFFFF            <1>       call   change_current_directory
0000AF3C 72A3                  <1>       jc     short msftdf_dsfde_error_retn
                               <1> 
                               <1> ;msftdf_dsfde_sf_change_prompt_dir_string:
                               <1> ;     call   change_prompt_dir_string
                               <1> 
                               <1> msftdf_dsfde_find_file:
0000AF3E BE[2A640100]          <1>       mov    esi, SourceFile_Name  ; Offset 66
0000AF43 668B460E              <1>       mov    ax, [esi+14] ; 80 -> SourceFile_AttributesMask
0000AF47 E817D3FFFF            <1>       call   find_first_file
0000AF4C 7293                  <1>       jc     short msftdf_dsfde_error_retn
                               <1> 
                               <1> msftdf_dsfde_delete_direntry:
0000AF4E 8B35[EC640100]        <1>       mov    esi, [msftdf_drv_offset]
                               <1> 
0000AF54 807E0300              <1>       cmp    byte [esi+LD_FATType], 0
0000AF58 770A                  <1>       ja     short msftdf_delete_FAT_direntry
                               <1> 
0000AF5A 30DB                  <1>       xor    bl, bl
                               <1>       ; BL = 0 -> File
                               <1>       ; EDI -> Directory buffer entry offset/address
0000AF5C E8E40B0000            <1>       call   delete_fs_directory_entry
0000AF61 7315                  <1>       jnc    short msftdf_dsfde_restore_current_dir_2
0000AF63 C3                    <1>       retn
                               <1> 
                               <1> msftdf_delete_FAT_direntry:
0000AF64 8A1D[F1620100]        <1>       mov    bl, [FindFile_LongNameEntryLength]
0000AF6A 668B0D[1C630100]      <1>       mov    cx, [FindFile_DirEntryNumber]
                               <1>       ; ESI = Logical DOS drive description table address
                               <1>       ; EDI = Directory buffer entry offset/address
0000AF71 E89AFCFFFF            <1>       call   delete_directory_entry
0000AF76 721C                  <1>       jc     short msftdf_retn
                               <1> 
                               <1> msftdf_dsfde_restore_current_dir_2:
0000AF78 803D[D30C0100]00      <1>       cmp    byte [Restore_CDIR], 0
0000AF7F 7607                  <1>       jna    short msftdf_new_dir_fcluster_retn
                               <1>       ;mov   esi, [msftdf_drv_offset]
0000AF81 E8F1BFFFFF            <1>       call   restore_current_directory
0000AF86 720C                  <1>       jc     short msftdf_retn
                               <1> 
                               <1> msftdf_new_dir_fcluster_retn:
0000AF88 31C9                  <1>       xor    ecx, ecx
0000AF8A A1[50650100]          <1>       mov    eax, [createfile_FFCluster]
0000AF8F BB[68640100]          <1>       mov    ebx, DestinationFile_Drv
                               <1> 
                               <1> msftdf_retn:
0000AF94 C3                    <1>       retn
```

```
                             <1> 
                             <1> 
                             <1> copy_source_file_to_destination_file:
                             <1>       ; 17/10/2016
                             <1>       ; 16/10/2016
                             <1>       ; 15/10/2016
                             <1>       ; 30/03/2016, 31/03/2016
                             <1>       ; 24/03/2016, 25/03/2016, 28/03/2016
                             <1>       ; 21/03/2016, 22/03/2016, 23/03/2016
                             <1>       ; 16/03/2016, 17/03/2016, 18/03/2016
                             <1>       ; 15/03/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 02/09/2011 (FILE.ASM 'copy_source_file_to_destination_file')
                             <1>       ; 01/08/2010 - 18/05/2011
                             <1>       ;
                             <1>       ;   Command Interpreter phase 1 enter ->
                             <1>       ;           AL = 1 -> Caller is command interpreter
                             <1>       ;           AL = 2 -> The second call, re-enter/continue
                             <1>       ;   Phase 1 -> Check source file
                             <1>       ;              'found' is required
                             <1>       ;   Phase 2 -> Check destination file,
                             <1>       ;              save 'found' or 'not found' status
                             <1>       ;              'permission denied' error will be return
                             <1>       ;              if attributes have not for ordinary file
                             <1>       ;              without readonly attribute
                             <1>       ;   Command Interpreter phase 1 return ->
                             <1>       ;              DH = Source file attributes
                             <1>       ;              DL = Destination file found status
                             <1>       ;              EAX = 0
                             <1>       ;   Command Interpreter phase 2 enter ->
                             <1>       ;              AL = 2 -> Continue from the last position
                             <1>       ;              AH =
                             <1>       ;   Phase 3 -> Load source file or use read/write cluster method
                             <1>       ;   Phase 4 -> Create destination file if it is not found
                             <1>       ;   Phase 5 -> Open destination file
                             <1>       ;   Phase 6 -> Read from source and write to destination
                             <1>       ;   Phase 7 -> Unload source file, if it is loaded at memory
                             <1>       ;       cf = 1 causes to return before the phase 7
                             <1>       ;              but loaded file will be unloaded
                             <1>       ;             (allocated memory block will be deallocated)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;       ESI = Source File Pathname (Asciiz)
                             <1>       ;        EDI = Destination File Pathname (Asciiz)
                             <1>       ;        AL = 0 --> Interrupt (System call)
                             <1>       ;        AL > 0 --> Command Interpreter (Question)
                             <1>       ;        AL = 1 --> Question Phase
                             <1>       ;        AL = 2 --> Progress Phase
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      cf = 0 -> OK
                             <1>       ;      EAX = Destination file first cluster
                             <1>       ;
                             <1>       ;        CL > 0 if there is file reading error before EOF
                             <1>       ;             (incomplete copy)
                             <1>       ;        CH > 0 if file is (full) loaded at memory
                             <1>       ;
                             <1>       ;      cf = 1 -> Error code in AL (EAX)
                             <1>       ;
                             <1>       ; (EBX, ECX, ESI, EDI register contents will be changed)
                             <1> 
                             <1> 
0000AF95 3C02                <1>       cmp    al, 2
0000AF97 0F845A020000        <1>       je     csftdf2_check_cdrv
                             <1> 
                             <1> ; Phase 1
                             <1> 
0000AF9D A2[10650100]        <1>       mov    byte [copy_cmd_phase], al
                             <1> 
0000AFA2 57                  <1>       push   edi ; *
                             <1> 
                             <1> csftdf_parse_sf_path:
0000AFA3 BF[E8630100]        <1>       mov    edi, SourceFile_Drv
0000AFA8 E887F4FFFF          <1>       call   parse_path_name
0000AFAD 721C                <1>       jc     short csftdf_parse_sf_path_failed
                             <1> 
                             <1> csftdf_parse_df_path:
0000AFAF 5E                  <1>       pop    esi ; * (pushed edi)
                             <1> 
                             <1> csftdf_sf_check_filename_exists:
0000AFB0 803D[2A640100]21    <1>       cmp    byte [SourceFile_Name], 21h
0000AFB7 7215                <1>       jb     short csftdf_sf_file_not_found_error
                             <1> 
0000AFB9 BF[68640100]        <1>       mov    edi, DestinationFile_Drv
0000AFBE E871F4FFFF          <1>       call   parse_path_name
0000AFC3 7310                <1>       jnc    short csftdf_check_sf_cdrv
                             <1> 
0000AFC5 3C01                <1>       cmp    al, 1 ; File or directory name is not existing
0000AFC7 760C                <1>       jna    short csftdf_check_sf_cdrv
                             <1> 
                             <1> csftdf_parse_df_path_failed:
0000AFC9 F9                  <1>       stc
                             <1> csftdf_sf_error_retn:
0000AFCA C3                  <1>       retn
                             <1> 
                             <1> csftdf_parse_sf_path_failed:
0000AFCB 5F                  <1>       pop    edi ; *
0000AFCC EBFC                <1>       jmp    short csftdf_sf_error_retn
                             <1> 
                             <1> csftdf_sf_file_not_found_error:
0000AFCE B802000000          <1>       mov    eax, 2 ; File not found
0000AFD3 EBF5                <1>       jmp    short csftdf_sf_error_retn
                             <1> 
                             <1> csftdf_check_sf_cdrv:
```

```
0000AFD5 8A3D[FE580100]      <1>       mov    bh, [Current_Drv]
                             <1> 
0000AFDB 883D[13650100]      <1>       mov    [csftdf_cdrv], bh ; 23/03/2016
                             <1> 
0000AFE1 8A15[E8630100]      <1>       mov    dl, [SourceFile_Drv]
0000AFE7 38FA                <1>       cmp    dl, bh ; byte [Current_Drv]
0000AFE9 7407                <1>       je     short csftdf_sf_check_directory
                             <1> 
0000AFEB E8D0BEFFFF          <1>       call   change_current_drive
0000AFF0 72D8                <1>       jc     short csftdf_sf_error_retn
                             <1> 
                             <1> csftdf_sf_check_directory:
0000AFF2 BE[E9630100]        <1>       mov    esi, SourceFile_Directory
0000AFF7 803E20              <1>       cmp    byte [esi], 20h
0000AFFA 760F                <1>       jna    short csftdf_find_sf
                             <1> 
                             <1> csftdf_sf_change_directory:
0000AFFC FE05[D30C0100]      <1>       inc    byte [Restore_CDIR]
0000B002 30E4                <1>       xor    ah, ah ; CD_COMMAND sign -> 0
0000B004 E815EEFFFF          <1>       call   change_current_directory
0000B009 72BF                <1>       jc     short csftdf_sf_error_retn
                             <1> 
                             <1> ;csftdf_sf_change_prompt_dir_string:
                             <1> ;     call   change_prompt_dir_string
                             <1> 
                             <1> csftdf_find_sf:
0000B00B BE[2A640100]        <1>       mov    esi, SourceFile_Name
0000B010 66B80018            <1>       mov    ax, 1800h ; Except volume label and dirs
0000B014 E84AD2FFFF          <1>       call   find_first_file
0000B019 72AF                <1>       jc     short csftdf_sf_error_retn
                             <1> 
                             <1> csftdf_sf_ambgfn_check:
0000B01B 6621D2              <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
0000B01E 7407                <1>       jz     short csftdf_sf_found
                             <1> 
                             <1> csftdf_ambiguous_file_name_error:
0000B020 B802000000          <1>       mov    eax, 2 ; File not found error
0000B025 F9                  <1>       stc
0000B026 C3                  <1>       retn
                             <1> 
                             <1> csftdf_sf_found:
0000B027 A3[14650100]        <1>       mov    [csftdf_filesize], eax
                             <1> 
0000B02C 09C0                <1>       or     eax, eax
0000B02E 7507                <1>       jnz    short csftdf_set_source_file_direntry
                             <1> 
                             <1> csftdf_sf_file_size_zero:
0000B030 B814000000          <1>       mov    eax, 20 ; TRDOS zero length (file size) error
0000B035 F9                  <1>       stc
0000B036 C3                  <1>       retn
                             <1> 
                             <1> csftdf_set_source_file_direntry:
0000B037 BE[F4620100]        <1>       mov    esi, FindFile_DirEntry
0000B03C BF[3A640100]        <1>       mov    edi, SourceFile_DirEntry
0000B041 B908000000          <1>       mov    ecx, 8
0000B046 F3A5                <1>       rep    movsd
                             <1> 
                             <1> csftdf_sf_restore_cdrv:
                             <1>       ; 22/03/2016
0000B048 8A15[13650100]      <1>       mov    dl, [csftdf_cdrv]
0000B04E 3A15[FE580100]      <1>       cmp    dl, [Current_Drv]
0000B054 7407                <1>       je     short csftdf_sf_restore_cdir
0000B056 E865BEFFFF          <1>       call   change_current_drive
0000B05B 724F                <1>       jc     short csftdf_df_error_retn ; 30/03/2016
                             <1> 
                             <1> csftdf_sf_restore_cdir:
0000B05D 803D[D30C0100]00    <1>       cmp    byte [Restore_CDIR], 0
0000B064 7612                <1>       jna    short csftdf_df_check_filename_exists
0000B066 29C0                <1>       sub    eax, eax
0000B068 BE00010900          <1>       mov    esi, Logical_DOSDisks
0000B06D 88D4                <1>       mov    ah, dl ; byte [csftdf_cdrv]
0000B06F 01C6                <1>       add    esi, eax
0000B071 E801BFFFFF          <1>       call   restore_current_directory
0000B076 7234                <1>       jc     short csftdf_df_error_retn
                             <1> 
                             <1> csftdf_df_check_filename_exists:
0000B078 803D[AA640100]20    <1>       cmp    byte [DestinationFile_Name], 20h
0000B07F 7716                <1>       ja     short csftdf_check_df_cdrv
                             <1> 
                             <1> csftdf_copy_sf_name:
0000B081 BF[AA640100]        <1>       mov    edi, DestinationFile_Name
0000B086 BE[2A640100]        <1>       mov    esi, SourceFile_Name
0000B08B B10C                <1>       mov    cl, 12
                             <1> 
                             <1> csftdf_df_copy_sf_name_loop:
0000B08D AC                  <1>       lodsb
0000B08E AA                  <1>       stosb
0000B08F 08C0                <1>       or     al, al
0000B091 7404                <1>       jz     short csftdf_check_df_cdrv
0000B093 FEC9                <1>       dec    cl
0000B095 75F6                <1>       jnz    csftdf_df_copy_sf_name_loop
                             <1> 
                             <1> csftdf_check_df_cdrv:
0000B097 8A15[68640100]      <1>       mov    dl, [DestinationFile_Drv]
0000B09D 3A15[FE580100]      <1>       cmp    dl, [Current_Drv]
0000B0A3 7408                <1>       je     short csftdf_df_check_directory
                             <1> 
0000B0A5 E816BEFFFF          <1>       call   change_current_drive
0000B0AA 7301                <1>       jnc    short csftdf_df_check_directory
                             <1> 
                             <1> csftdf_df_error_retn:
0000B0AC C3                  <1>       retn
                             <1> 
                             <1> csftdf_df_check_directory:
0000B0AD BE[69640100]        <1>       mov    esi, DestinationFile_Directory
0000B0B2 803E20              <1>         cmp     byte [esi], 20h
0000B0B5 760F                <1>       jna    short csftdf_find_df
                             <1> 
                             <1> csftdf_df_change_directory:
0000B0B7 FE05[D30C0100]      <1>       inc    byte [Restore_CDIR]
0000B0BD 28E4                <1>       sub    ah, ah ; CD_COMMAND sign -> 0
0000B0BF E85AEDFFFF          <1>       call   change_current_directory
0000B0C4 72E6                <1>       jc     short csftdf_df_error_retn
                             <1> 
                             <1> ;csftdf_df_change_prompt_dir_string:
                             <1> ;     call   change_prompt_dir_string
                             <1> 
                             <1> csftdf_find_df:
                             <1>       ; 23/03/2016
0000B0C6 29DB                <1>       sub    ebx, ebx
0000B0C8 8A3D[68640100]      <1>       mov    bh,  [DestinationFile_Drv]
0000B0CE 81C300010900        <1>       add    ebx, Logical_DOSDisks
0000B0D4 891D[40650100]      <1>       mov    [csftdf_df_drv_dt], ebx
                             <1> 
0000B0DA BE[AA640100]        <1>       mov    esi, DestinationFile_Name
0000B0DF 6631C0              <1>       xor    ax, ax
                             <1>              ; DestinationFile_AttributesMask -> any/zero
0000B0E2 E87CD1FFFF          <1>       call   find_first_file
0000B0E7 7218                <1>       jc     short csftdf_df_check_error_code
                             <1> 
                             <1> csftdf_df_ambgfn_check:
0000B0E9 6609D2              <1>       or     dx, dx ; Ambiguous filename chars used sign (DX>0)
0000B0EC 752A                <1>       jnz    short csftdf_df_error_inv_fname
                             <1> 
                             <1> csftdf_df_found:
0000B0EE C605[12650100]01    <1>       mov    byte [DestinationFileFound], 1
                             <1>       ; 17/10/2016 (cl -> bl)
0000B0F5 80E31F              <1>       and    bl, 1Fh ; Attributes, D-V-S-H-R
0000B0F8 745F                <1>       jz     short csftdf_df_save_first_cluster
                             <1> 
                             <1> csftdf_df_permission_denied_retn:
0000B0FA B805000000          <1>       mov    eax, 05h ; Access/Permission denied.
                             <1> csftdf_df_error_stc_retn:
0000B0FF F9                  <1>       stc
0000B100 C3                  <1>       retn
                             <1> 
                             <1> csftdf_df_check_error_code:
                             <1>       ;cmp   eax, 2
0000B101 3C02                <1>       cmp    al, 2
0000B103 75FA                <1>       jne    short csftdf_df_error_stc_retn
                             <1> 
0000B105 C605[12650100]00    <1>       mov    byte [DestinationFileFound], 0
                             <1> 
                             <1>       ; 15/10/2016
0000B10C BE[E4620100]        <1>       mov    esi, FindFile_Name ; *
0000B111 E80BD5FFFF          <1>       call   check_filename
0000B116 7307                <1>       jnc    short csftdf_df_valid_fname
                             <1> csftdf_df_error_inv_fname: ; 'invalid file name !'
0000B118 B81A000000          <1>       mov    eax, ERR_INV_FILE_NAME  ; 26
0000B11D F9                  <1>       stc
0000B11E C3                  <1>       retn
                             <1> 
                             <1> csftdf_df_valid_fname:
                             <1>       ; 21/03/2016
                             <1>       ; (Capitalized file name)
                             <1>       ;mov   esi, FindFile_Name ; * ; 15/10/2016
0000B11F BF[AA640100]        <1>       mov    edi, DestinationFile_Name
0000B124 A5                  <1>       movsd
0000B125 A5                  <1>       movsd
0000B126 A5                  <1>       movsd
                             <1>       ;movsb
                             <1> 
                             <1> csftdf_check_disk_free_size_0:
0000B127 A1[56640100]        <1>       mov    eax, [SourceFile_DirEntry+DirEntry_FileSize]
                             <1> 
                             <1> csftdf_check_disk_free_size_1:
                             <1>       ;sub   ebx, ebx
                             <1>       ;mov   esi, Logical_DOSDisks
                             <1>       ;mov   bh,  [DestinationFile_Drv]
                             <1>       ;add   esi, ebx
                             <1> 
0000B12C 8B35[40650100]      <1>       mov    esi, [csftdf_df_drv_dt] ; 23/03/2016
                             <1> 
0000B132 0FB74E11            <1>       movzx  ecx, word [esi+LD_BPB+BytesPerSec] ; 17, LD_BPB + 0Bh
0000B136 01C8                <1>       add    eax, ecx
0000B138 48                  <1>       dec    eax  ; file size (additional bytes) + 511 (round up)
                             <1> csftdf_check_disk_free_size_3: ; 16/03/2016
0000B139 29D2                <1>       sub    edx, edx
0000B13B F7F1                <1>       div    ecx ; bytes per sector
                             <1> 
                             <1> csftdf_check_disk_free_size:
0000B13D 3B4674              <1>       cmp    eax, [esi+LD_FreeSectors]
0000B140 0F8294000000        <1>         jb      csftdf_check_disk_free_size_ok
0000B146 770A                <1>       ja     short csftdf_df_insufficient_disk_space
                             <1> 
0000B148 807E0300            <1>       cmp    byte [esi+LD_FATType], 0 ; FS needs FDT sector also.
0000B14C 0F8788000000        <1>         ja      csftdf_check_disk_free_size_ok
                             <1> 
                             <1> csftdf_df_insufficient_disk_space:
0000B152 B827000000          <1>       mov    eax, 27h ; insufficient disk space
0000B157 EBA6                <1>       jmp    short csftdf_df_error_stc_retn
                             <1> 
                             <1> csftdf_df_save_first_cluster:
                             <1>       ; ESI = FindFile_DirEntry (for the old destination file)
                             <1>       ; EAX = Old destination file size
                             <1>       ; 24/03/2016
```

```
                               <1>       ; EDI = Directory entry address (within Dir Buffer boundaries)
0000B159 81EF00000800          <1>       sub    edi, Directory_Buffer  ; (<65536)
0000B15F 66C1EF05              <1>       shr    di, 5 ; Convert entry offset to entry index/number
0000B163 66893D[E2640100]      <1>       mov    [DestinationFile_DirEntryNumber], di ; (<2048)
                               <1>
                               <1> csftdf_df_check_sf_df_fcluster:
0000B16A 668B5614              <1>       mov    dx, [esi+DirEntry_FstClusHI]
0000B16E C1E210                <1>       shl    edx, 16
0000B171 668B561A              <1>       mov    dx, [esi+DirEntry_FstClusLO]
0000B175 8915[24650100]        <1>       mov    [csftdf_df_cluster], edx
                               <1> csftdf_df_check_sf_df_fcluster_1:
0000B17B 668B15[4E640100]      <1>       mov    dx, [SourceFile_DirEntry+DirEntry_FstClusHI]
0000B182 C1E210                <1>       shl    edx, 16
0000B185 668B15[54640100]      <1>       mov    dx, [SourceFile_DirEntry+DirEntry_FstClusLO]
0000B18C 3B15[24650100]        <1>       cmp    edx, [csftdf_df_cluster]
0000B192 7512                  <1>       jne    short csftdf_df_check_sf_df_fcluster_ok
                               <1> csftdf_df_check_sf_df_drv:
0000B194 8A15[E8630100]        <1>       mov    dl, [SourceFile_Drv]
0000B19A 3A15[68640100]        <1>       cmp    dl, [DestinationFile_Drv]
0000B1A0 7504                  <1>       jne    short csftdf_df_check_sf_df_fcluster_ok
                               <1>
                               <1>       ; source and destination files are same !
                               <1>       ; (they have same first cluster value on same logical disk)
                               <1>
0000B1A2 31C0                  <1>       xor    eax, eax ; mov eax, 0 -> Bad command or file name !
0000B1A4 F9                    <1>       stc
0000B1A5 C3                    <1>       retn
                               <1>
                               <1> csftdf_df_check_sf_df_fcluster_ok:
                               <1> csftdf_df_move_findfile_struct:
                               <1>       ; mov  esi, FindFile_DirEntry
0000B1A6 BF[BA640100]          <1>       mov    edi, DestinationFile_DirEntry
0000B1AB B908000000            <1>       mov    ecx, 8
0000B1B0 F3A5                  <1>       rep    movsd
                               <1>
                               <1> csftdf_check_disk_free_size_2:
0000B1B2 89C2                  <1>       mov    edx, eax ; Old destination file size
                               <1>
                               <1>       ;mov   eax, [SourceFile_DirEntry+DirEntry_FileSize]
0000B1B4 A1[14650100]          <1>       mov    eax, [csftdf_filesize] ; 23/03/2016
                               <1>
                               <1>       ;;sub  ecx, ecx ; 0
                               <1>       ;mov   esi, Logical_DOSDisks
                               <1>       ;mov   ch,  [DestinationFile_Drv]
                               <1>       ;add   esi, ecx
                               <1>       ;
                               <1>       ;mov   [csftdf_df_drv_dt], esi
                               <1>
0000B1B9 8B35[40650100]        <1>       mov    esi, [csftdf_df_drv_dt] ; 23/03/2016
                               <1>
0000B1BF 668B4E11              <1>       mov    cx, [esi+LD_BPB+BytesPerSec] ; 17, LD_BPB + 0Bh
0000B1C3 01CA                  <1>       add    edx, ecx ; + 512
0000B1C5 01C8                  <1>       add    eax, ecx ; + 512
0000B1C7 4A                    <1>       dec    edx ; old file size + 511 (round up)
0000B1C8 48                    <1>       dec    eax ; new file size + 511 (round up)
0000B1C9 F7D9                  <1>       neg    ecx ; -512 ; 0FFFFFE00h
0000B1CB 21CA                  <1>       and    edx, ecx ; = old sector count * 512
0000B1CD 21C8                  <1>       and    eax, ecx ; = new sector count * 512
                               <1>
0000B1CF 29D0                  <1>       sub    eax, edx ; new file size - old file size (on disk)
0000B1D1 7607                  <1>       jna    short csftdf_check_disk_free_size_ok
                               <1>
0000B1D3 F7D9                  <1>       neg    ecx ; 512 (bytes per sector) ; 200h
                               <1>       ; check free space for additional sectors
                               <1>       ; eax = number of additional sectors * bytes per sector
                               <1>       ; esi = Logical DOS drive number (of destination disk)
0000B1D5 E95FFFFFFF            <1>         jmp     csftdf_check_disk_free_size_3
                               <1>
                               <1> csftdf_check_disk_free_size_ok:
                               <1>       ; 18/03/2016
                               <1> csftdf_df_check_copy_cmd_phase:
0000B1DA A0[10650100]          <1>       mov    al, [copy_cmd_phase]
0000B1DF 3C01                  <1>       cmp    al, 1
0000B1E1 7514                  <1>       jne    short csftdf2_check_cdrv
                               <1>
0000B1E3 31C0                  <1>       xor    eax, eax
0000B1E5 A2[10650100]          <1>       mov    [copy_cmd_phase], al ; 0
                               <1>
0000B1EA 8A15[12650100]        <1>       mov    dl, [DestinationFileFound]
0000B1F0 8A35[45640100]        <1>       mov    dh, [SourceFile_DirEntry+11] ; Attributes
                               <1>
                               <1> csftdf_return:
0000B1F6 C3                    <1>       retn
                               <1>
                               <1> ; Phase 2
                               <1>
                               <1> csftdf2_check_cdrv:
                               <1>       ; 18/03/2016
                               <1>       ; Here, destination drive and directory are ready !
                               <1>       ; (checking/restoring is not needed)
                               <1>       ; (Since at the end of the phase 1)
                               <1>
                               <1> ;     mov    dl, [DestinationFile_Drv]
                               <1> ;     cmp    dl, [Current_Drv]
                               <1> ;     je     short csftdf2_df_check_directory
                               <1> ;
                               <1> ;     call   change_current_drive
                               <1> ;     jc     short csftdf2_read_error
                               <1> ;
                               <1> ;csftdf2_df_check_directory:
                               <1> ;     mov    esi, DestinationFile_Directory
                               <1> ;     cmp    byte [esi], 20h
                               <1> ;     jna    short csftdf2_df_check_found_or_not
```

```
                             <1> ;
                             <1> ;csftdf2_df_change_directory:
                             <1> ;     inc    byte [Restore_CDIR]
                             <1> ;     xor    ah, ah ; CD_COMMAND sign -> 0
                             <1> ;     call   change_current_directory
                             <1> ;     jc     short csftdf2_stc_return
                             <1> ;
                             <1> ;;csftdf2_df_change_prompt_dir_string:
                             <1> ;;    call   change_prompt_dir_string
                             <1>
                             <1> csftdf2_df_check_found_or_not:
                             <1>       ; 21/03/2016
0000B1F7 803D[12650100]00    <1>       cmp    byte [DestinationFileFound], 0
0000B1FE 7739                <1>       ja     short csftdf2_set_sf_percentage
                             <1>
                             <1> csftdf2_create_file:
0000B200 BE[AA640100]        <1>       mov    esi, DestinationFile_Name
0000B205 A1[14650100]        <1>       mov    eax, [csftdf_filesize]
0000B20A 30C9                <1>       xor    cl, cl ; 0
                             <1>
0000B20C 31DB                <1>       xor    ebx, ebx ; 0
0000B20E 4B                  <1>       dec    ebx ; 0FFFFFFFFh
                             <1>
                             <1>       ; INPUT ->
                             <1>       ;      EAX -> File Size
                             <1>       ;      ESI = ASCIIZ File name
                             <1>       ;       CL = File attributes
                             <1>       ;      EBX = FFFFFFFFh -> empty file sign for FAT fs
                             <1>       ;      EBX <> FFFFFFFFh -> use file size for FAT fs
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      EAX = New file's first cluster
                             <1>       ;      ESI = Logical Dos Drv Descr. Table Addr.
                             <1>       ;      EBX = CreateFile_Size address
                             <1>       ;      ECX = Sectors per cluster (<256)
                             <1>       ;      EDX = Directory Entry Index/Number (<65536)
                             <1>       ;
                             <1>       ;      cf = 1 -> error code in AL (EAX)
                             <1>
0000B20F E8EC050000          <1>       call   create_file
                             <1>       ;pop   esi
0000B214 0F82A3050000        <1>         jc      csftdf2_rw_error
                             <1>
                             <1> csftdf2_create_file_OK:
0000B21A A3[24650100]        <1>       mov    [csftdf_df_cluster], eax
                             <1>
                             <1>       ; 24/03/2016
0000B21F 668915[E2640100]    <1>       mov    [DestinationFile_DirEntryNumber], dx
                             <1>
                             <1>       ; 21/03/2016
0000B226 BE00000800          <1>       mov    esi, Directory_Buffer
0000B22B C1E205              <1>       shl    edx, 5 ; 32 * index number
0000B22E 01D6                <1>       add    esi, edx
0000B230 BF[BA640100]        <1>       mov    edi, DestinationFile_DirEntry
0000B235 B108                <1>       mov    cl, 8 ; 32 bytes
0000B237 F3A5                <1>       rep    movsd
                             <1>
                             <1> csftdf2_set_sf_percentage:
                             <1>       ; 17/03/2016
0000B239 31C0                <1>       xor    eax, eax
0000B23B A2[38650100]        <1>       mov    [csftdf_percentage], al ; 0, reset
                             <1>
0000B240 A3[30650100]        <1>       mov    [csftdf_sf_rbytes], eax ; 0, reset
0000B245 A3[34650100]        <1>       mov    [csftdf_df_wbytes], eax ; 0, reset
                             <1>
0000B24A 8A25[E8630100]      <1>       mov    ah, [SourceFile_Drv]
0000B250 BE00010900          <1>       mov    esi, Logical_DOSDisks
0000B255 01C6                <1>       add    esi, eax
                             <1>
0000B257 8935[3C650100]      <1>       mov    [csftdf_sf_drv_dt], esi ; 23/03/2016
                             <1>
0000B25D 668B15[4E640100]    <1>       mov    dx, [SourceFile_DirEntry+DirEntry_FstClusHI]
0000B264 C1E210              <1>       shl    edx, 16
0000B267 668B15[54640100]    <1>       mov    dx, [SourceFile_DirEntry+DirEntry_FstClusLO]
0000B26E 8915[20650100]      <1>       mov    [csftdf_sf_cluster], edx
                             <1>
                             <1>       ; 16/03/2016
                             <1>       ; Note: Singlix FS boot sector parameters (for cluster
                             <1>       ;      related calculations) has same offset
                             <1>       ;      values from LD_BPB as in FAT file system.
                             <1>       ;      [esi+LD_BPB+SecPerClust] is 1 for Singlix FS.
                             <1>       ;
0000B274 0FB64E13            <1>       movzx ecx, byte [esi+LD_BPB+SecPerClust]
0000B278 880D[66640100]      <1>       mov    [SourceFile_SecPerClust], cl
                             <1>
                             <1>       ; 17/03/2016
0000B27E 386E03              <1>       cmp    [esi+LD_FATType], ch ; 0
0000B281 7707                <1>       ja     short csftdf2_set_sf_percent_rsize1
                             <1>
0000B283 B800000100          <1>       mov    eax, 65536 ; read/write buffer size for Singlix FS
0000B288 EB06                <1>       jmp    short csftdf2_set_sf_percent_rsize2
                             <1>
                             <1> csftdf2_set_sf_percent_rsize1:
0000B28A 668B4611            <1>       mov    ax, [esi+LD_BPB+BytesPerSec]
0000B28E F7E1                <1>       mul    ecx
                             <1>       ;sub   edx, edx
                             <1> csftdf2_set_sf_percent_rsize2:
0000B290 A3[28650100]        <1>       mov    [csftdf_r_size], eax
                             <1>
                             <1> csftdf2_set_df_percentage:
                             <1>       ;sub   eax, eax
                             <1>       ;mov   ah, [DestinationFile_Drv]
                             <1>       ;mov   edi, Logical_DOSDisks
                             <1>       ;add   edi, eax
                             <1>       ;mov   [csftdf_df_drv_dt], edi ; 17/03/2016
                             <1> 
0000B295 8B3D[40650100]      <1>       mov    edi, [csftdf_df_drv_dt] ; 23/03/2016
                             <1> 
                             <1>       ; 16/03/2016
                             <1>       ; Note: Singlix FS boot sector parameters (for cluster
                             <1>       ;      related calculations) has same offset
                             <1>       ;      values from LD_BPB as in FAT file system.
                             <1>       ;      [edi+LD_BPB+SecPerClust] is 1 for Singlix FS.
                             <1>       ;
                             <1>       ;movzx ecx, byte [edi+LD_BPB+SecPerClust]
0000B29B 8A4F13              <1>       mov    cl, [edi+LD_BPB+SecPerClust]
0000B29E 880D[E6640100]      <1>       mov    [DestinationFile_SecPerClust], cl
                             <1> 
                             <1>       ; 17/03/2016
0000B2A4 386F03              <1>       cmp    [edi+LD_FATType], ch ; 0
0000B2A7 7707                <1>       ja     short csftdf2_set_df_percent_wsize1
                             <1> 
0000B2A9 B800000100          <1>       mov    eax, 65536 ; read/write buffer size for Singlix FS
0000B2AE EB06                <1>       jmp    short csftdf2_set_df_percent_wsize2
                             <1> 
                             <1> csftdf2_set_df_percent_wsize1:
0000B2B0 0FB74711            <1>       movzx  eax, word [edi+LD_BPB+BytesPerSec]
0000B2B4 F7E1                <1>       mul    ecx
                             <1>       ;sub   edx, edx
                             <1> csftdf2_set_df_percent_wsize2:
0000B2B6 A3[2C650100]        <1>       mov    [csftdf_w_size], eax
                             <1> 
0000B2BB A1[14650100]        <1>       mov    eax, [csftdf_filesize]
                             <1> 
0000B2C0 3D00000100          <1>       cmp    eax, 65536 ; 64KB   ; small file
0000B2C5 721F                <1>       jb     short csftdf2_load_file ; do not display percentage
                             <1> 
                             <1> csftdf2_reset_wf_percent_ptr_chk_64k:
0000B2C7 B201                <1>       mov    dl, 1 ; 25/03/2016
                             <1> 
0000B2C9 3D00000400          <1>       cmp    eax, 65536*4 ; 256KB
0000B2CE 7310                <1>       jnb    short csftdf2_enable_percentage_display ; big file
                             <1> 
                             <1>       ; 64-128KB file size for floppy disks
0000B2D0 3815[E8630100]      <1>       cmp    byte [SourceFile_Drv], dl ; 1 ; read from floppy disk ?
0000B2D6 7608                <1>       jna    short csftdf2_enable_percentage_display
                             <1> 
0000B2D8 3815[68640100]      <1>       cmp    byte [DestinationFile_Drv], dl ; 1 ; write to floppy disk ?
0000B2DE 7706                <1>       ja     short csftdf2_load_file
                             <1> 
                             <1> csftdf2_enable_percentage_display:
0000B2E0 8815[38650100]      <1>       mov    [csftdf_percentage], dl ; 1
                             <1> 
                             <1> csftdf2_load_file:
                             <1>       ; 13/05/2016
                             <1>       ; 19/03/2016
                             <1>       ; 18/03/2016
                             <1>       ; 17/03/2016
0000B2E6 B40F                <1>       mov    ah, 0Fh
0000B2E8 E8AD61FFFF          <1>       call   _int10h
                             <1>       ; 13/05/2016
0000B2ED 883D[39650100]      <1>       mov    [csftdf_videopage], bh ; active video page
0000B2F3 B403                <1>       mov    ah, 03h
0000B2F5 E8A061FFFF          <1>       call   _int10h
0000B2FA 668915[3A650100]    <1>       mov    [csftdf_cursorpos], dx
                             <1> 
0000B301 29C0                <1>       sub    eax, eax
0000B303 A2[11650100]        <1>       mov    [csftdf_rw_err], al ; 0
                             <1> 
                             <1> ; ///
                             <1> csftdf_sf_amb: ; 15/03/2016
0000B308 8B0D[14650100]      <1>       mov    ecx, [csftdf_filesize]    ; 23/03/2016
                             <1> 
                             <1>       ; TRDOS 386 (TRDOS v2.0)
                             <1>       ; Allocate contiguous memory block for loading the file
                             <1> 
                             <1>       ;mov   ecx, [SourceFile_DirEntry+DirEntry_FileSize]
                             <1> 
                             <1>       ;sub   eax, eax ; First free memory aperture
                             <1> 
                             <1>       ; eax = 0 (Allocate memory from the beginning)
                             <1>       ; ecx = File (Allocation) size in bytes
                             <1> 
0000B30E E811A1FFFF          <1>       call   allocate_memory_block
0000B313 7304                <1>       jnc    short loc_check_sf_save_loading_parms
                             <1> 
0000B315 29C0                <1>       sub    eax, eax
0000B317 29C9                <1>       sub    ecx, ecx
                             <1> 
                             <1> loc_check_sf_save_loading_parms:
0000B319 A3[18650100]        <1>       mov    [csftdf_sf_mem_addr], eax ; loading address
0000B31E 890D[1C650100]      <1>       mov    [csftdf_sf_mem_bsize], ecx ; block size
                             <1> ; ///
                             <1>       ; 19/03/2016
0000B324 8B35[3C650100]      <1>       mov    esi, [csftdf_sf_drv_dt] ; logical dos drv desc. tbl.
                             <1> 
                             <1>       ; 17/03/2016
0000B32A 09C0                <1>       or     eax, eax ; contiguous free memory block address
0000B32C 0F845B010000        <1>         jz      csftdf2_read_sf_cluster
                             <1> 
                             <1>       ; 18/03/2016
0000B332 8B1D[18650100]      <1>       mov    ebx, [csftdf_sf_mem_addr] ; memory block address
                             <1> 
0000B338 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
0000B33C 0F8605020000        <1>         jna     csftdf2_load_fs_file
                             <1> 
```

```
                              <1> csftdf2_load_fat_file:
0000B342 53                   <1>       push   ebx ; *
                              <1>
                              <1> csftdf2_load_fat_file_next:
0000B343 BE[23130100]         <1>       mov    esi, msg_reading
0000B348 E810B0FFFF           <1>       call   print_msg
                              <1>
0000B34D 803D[38650100]00     <1>       cmp    byte [csftdf_percentage], 0
0000B354 7605                 <1>       jna    short csftdf2_load_fat_file_1
                              <1>
0000B356 E87C000000           <1>       call   csftdf2_print_percentage ; 19/03/2016
                              <1>
                              <1> csftdf2_load_fat_file_1:
0000B35B 8B35[3C650100]       <1>       mov    esi, [csftdf_sf_drv_dt]
0000B361 5B                   <1>       pop    ebx ; *
                              <1>
                              <1> csftdf2_load_fat_file_2:
0000B362 E8B8000000           <1>       call   csftdf2_read_fat_file_sectors ; 19/03/2016
0000B367 0F8250040000         <1>         jc      csftdf2_rw_error ; eocc! or disk error!
                              <1>
0000B36D 09D2                 <1>       or     edx, edx ; edx > 0 -> EOF
0000B36F 7520                 <1>       jnz    short csftdf2_load_fat_file_ok
                              <1>
0000B371 803D[38650100]00     <1>       cmp    byte [csftdf_percentage], 0
0000B378 76E8                 <1>       jna    short csftdf2_load_fat_file_2
                              <1>
0000B37A 53                   <1>       push   ebx ; *
                              <1>
                              <1>       ; Set cursor position
                              <1>       ; AH= 02h, BH= Page Number, DH= Row, DL= Column
0000B37B 8A3D[39650100]       <1>       mov    bh, [csftdf_videopage]
0000B381 668B15[3A650100]     <1>       mov    dx, [csftdf_cursorpos]
0000B388 B402                 <1>       mov    ah, 2
0000B38A E80B61FFFF           <1>       call   _int10h
0000B38F EBB2                 <1>       jmp    short csftdf2_load_fat_file_next
                              <1>
                              <1> csftdf2_load_fat_file_ok:
0000B391 803D[38650100]00     <1>       cmp    byte [csftdf_percentage], 0
0000B398 0F8651020000         <1>         jna     csftdf2_save_file ; 25/03/2016
                              <1>
                              <1>       ; "Reading... 100%"
0000B39E BF[3B130100]         <1>       mov    edi, percentagestr
0000B3A3 B031                 <1>       mov    al, '1'
0000B3A5 AA                   <1>       stosb
0000B3A6 B030                 <1>       mov    al, '0'
0000B3A8 AA                   <1>       stosb
0000B3A9 AA                   <1>       stosb
                              <1>
0000B3AA 8A3D[39650100]       <1>       mov    bh, [csftdf_videopage]
0000B3B0 668B15[3A650100]     <1>       mov    dx, [csftdf_cursorpos]
0000B3B7 B402                 <1>       mov    ah, 2
0000B3B9 E8DC60FFFF           <1>       call   _int10h
                              <1>
0000B3BE BE[23130100]         <1>       mov    esi, msg_reading
0000B3C3 E895AFFFFF           <1>       call   print_msg
                              <1>
0000B3C8 BE[3B130100]         <1>       mov    esi, percentagestr
0000B3CD E88BAFFFFF           <1>       call   print_msg
                              <1>
0000B3D2 E918020000           <1>         jmp     csftdf2_save_file ; 25/03/2016
                              <1>
                              <1> csftdf2_print_percentage:
                              <1>       ; 09/12/2017
                              <1>       ; 19/03/2016
                              <1>       ; 18/03/2016
0000B3D7 B020                 <1>       mov    al, 20h
0000B3D9 BF[3B130100]         <1>       mov    edi, percentagestr
0000B3DE AA                   <1>       stosb
0000B3DF AA                   <1>       stosb
0000B3E0 A1[30650100]         <1>       mov    eax, [csftdf_sf_rbytes]
0000B3E5 BA64000000           <1>       mov    edx, 100
0000B3EA F7E2                 <1>       mul    edx
0000B3EC 8B0D[14650100]       <1>       mov    ecx, [csftdf_filesize]
0000B3F2 F7F1                 <1>       div    ecx
0000B3F4 B10A                 <1>       mov    cl, 10
0000B3F6 F6F1                 <1>       div    cl
0000B3F8 80C430               <1>       add    ah, '0'
0000B3FB 8827                 <1>       mov    [edi], ah
0000B3FD 20C0                 <1>       and    al, al
0000B3FF 740A                 <1>       jz     short csftdf2_print_percent_1
0000B401 4F                   <1>       dec    edi
                              <1>       ;cbw
0000B402 28E4                 <1>       sub    ah, ah ; 09/12/2017
0000B404 F6F1                 <1>       div    cl
0000B406 80C430               <1>       add    ah, '0'
0000B409 8827                 <1>       mov    [edi], ah
                              <1>       ;and   al, al
                              <1>       ;jz    short csftdf2_print_percent_1
                              <1>       ;dec   edi
                              <1>       ;mov   [edi], '1' ; 100%
                              <1>
                              <1> csftdf2_print_percent_1:
0000B40B BE[3B130100]         <1>       mov    esi, percentagestr
                              <1>       ;call  print_msg
                              <1>       ;retn
0000B410 E948AFFFFF           <1>       jmp    print_msg
                              <1>
                              <1> csftdf2_read_file_sectors:
                              <1>       ; 19/03/2016
0000B415 807E0300             <1>       cmp    byte [esi+LD_FATType], 0
0000B419 0F8627070000         <1>         jna     csftdf2_read_fs_file_sectors
                              <1>
                              <1> csftdf2_read_fat_file_sectors:
```

```
                             <1>       ; 19/03/2016
                             <1>       ; 18/03/2016
                             <1>       ; return:
                             <1>       ;   CF = 0 & EDX > 0 -> END OF FILE
                             <1>       ;   CF = 0 & EDX = 0 -> not EOF
                             <1>       ;   CF = 1 -> read error (error code in AL)
                             <1>
                             <1> csftdf2_read_fat_file_secs_0:
0000B41F 8B15[14650100]      <1>       mov    edx, [csftdf_filesize]
0000B425 2B15[30650100]      <1>       sub    edx, [csftdf_sf_rbytes]
0000B42B 3B15[28650100]      <1>       cmp    edx, [csftdf_r_size]
0000B431 7306                <1>       jnb    short csftdf2_read_fat_file_secs_1
0000B433 8915[28650100]      <1>       mov    [csftdf_r_size], edx
                             <1>
                             <1> csftdf2_read_fat_file_secs_1:
0000B439 A1[28650100]        <1>       mov    eax, [csftdf_r_size]
0000B43E 29D2                <1>       sub    edx, edx
0000B440 0FB74E11            <1>       movzx ecx, word [esi+LD_BPB+BytesPerSec]
0000B444 01C8                <1>       add    eax, ecx
0000B446 48                  <1>       dec    eax
0000B447 F7F1                <1>       div    ecx
0000B449 89C1                <1>       mov    ecx, eax ; sector count
0000B44B A1[20650100]        <1>       mov    eax, [csftdf_sf_cluster]
                             <1>
                             <1>       ; EBX = memory block address (current)
                             <1>
0000B450 E821090000          <1>       call   read_fat_file_sectors
0000B455 7235                <1>       jc     short csftdf2_read_fat_file_secs_3
                             <1>
                             <1>       ; EBX = next memory address
                             <1>
0000B457 A1[30650100]        <1>       mov    eax, [csftdf_sf_rbytes]
0000B45C 0305[28650100]      <1>       add    eax, [csftdf_r_size]
0000B462 8B15[14650100]      <1>       mov    edx, [csftdf_filesize]
0000B468 39D0                <1>       cmp    eax, edx
0000B46A 7320                <1>       jnb    short csftdf2_read_fat_file_secs_3 ; edx > 0
0000B46C A3[30650100]        <1>       mov    [csftdf_sf_rbytes], eax
                             <1>
0000B471 53                  <1>       push   ebx ; *
                             <1>       ; get next cluster (csftdf_r_size! bytes)
0000B472 A1[20650100]        <1>       mov    eax, [csftdf_sf_cluster]
0000B477 E8CC060000          <1>       call   get_next_cluster
0000B47C 5B                  <1>       pop    ebx ; *
0000B47D 7306                <1>       jnc    short csftdf2_read_fat_file_secs_2
                             <1>
                             <1>       ; 15/10/2016
                             <1>       ;Disk read error instad of drv not ready err
0000B47F B811000000          <1>       mov    eax, 17 ; Read error !
0000B484 C3                  <1>       retn
                             <1>
                             <1> csftdf2_read_fat_file_secs_2:
0000B485 29D2                <1>       sub    edx, edx ; 0
0000B487 A3[20650100]        <1>       mov    [csftdf_sf_cluster], eax ; next cluster
                             <1>
                             <1> csftdf2_read_fat_file_secs_3:
0000B48C C3                  <1>       retn
                             <1>
                             <1> csftdf2_read_sf_cluster:
                             <1>       ; 19/03/2016
0000B48D BB00000700          <1>       mov    ebx, Cluster_Buffer ; buffer address (64KB)
                             <1>
0000B492 803D[38650100]00    <1>       cmp    byte [csftdf_percentage], 0
0000B499 760D                <1>       jna    short csftdf2_read_sf_clust_2
                             <1>
0000B49B 53                  <1>       push   ebx ; *
                             <1>
                             <1> csftdf2_read_sf_clust_next:
0000B49C E836FFFFFF          <1>       call   csftdf2_print_percentage
                             <1>
                             <1> csftdf2_read_sf_clust_0:
0000B4A1 8B35[3C650100]      <1>       mov    esi, [csftdf_sf_drv_dt]
                             <1> csftdf2_read_sf_clust_1:
0000B4A7 5B                  <1>       pop    ebx ; *
                             <1>
                             <1> csftdf2_read_sf_clust_2:
0000B4A8 89DA                <1>       mov    edx, ebx
0000B4AA 0315[28650100]      <1>       add    edx, [csftdf_r_size]
0000B4B0 81FA00000800        <1>       cmp    edx, Cluster_Buffer + 65536
0000B4B6 772F                <1>       ja     short csftdf2_write_df_cluster
                             <1>
0000B4B8 E858FFFFFF          <1>       call   csftdf2_read_file_sectors ; 19/03/2016
0000B4BD 0F8280020000        <1>         jc      csftdf2_save_fat_file_err2 ; eocc! or disk error!
                             <1>
0000B4C3 09D2                <1>       or     edx, edx ; edx > 0 -> EOF
0000B4C5 7520                <1>       jnz    short csftdf2_write_df_cluster
                             <1>
0000B4C7 803D[38650100]00    <1>       cmp    byte [csftdf_percentage], 0
0000B4CE 76D8                <1>       jna    short csftdf2_read_sf_clust_2
                             <1>
0000B4D0 53                  <1>       push   ebx ; *
                             <1>
                             <1>       ; Set cursor position
                             <1>       ; AH= 02h, BH= Page Number, DH= Row, DL= Column
0000B4D1 8A3D[39650100]      <1>       mov    bh, [csftdf_videopage]
0000B4D7 668B15[3A650100]    <1>       mov    dx, [csftdf_cursorpos]
0000B4DE B402                <1>       mov    ah, 2
0000B4E0 E8B55FFFFF          <1>       call   _int10h
0000B4E5 EBB5                <1>       jmp    short csftdf2_read_sf_clust_next
                             <1>
                             <1> csftdf2_write_df_cluster:
                             <1>       ; 19/03/2016
0000B4E7 8B35[40650100]      <1>       mov    esi, [csftdf_df_drv_dt]
0000B4ED BB00000700          <1>       mov    ebx, Cluster_Buffer ; buffer address (64KB)
```

```
                               <1>
                               <1> csftdf2_write_df_clust_next:
0000B4F2 E855000000            <1>       call   csftdf2_write_file_sectors ; 19/03/2016
0000B4F7 0F8246020000          <1>         jc      csftdf2_save_fat_file_err2 ; eocc! or disk error!
                               <1>
0000B4FD 09D2                  <1>       or     edx, edx ; edx > 0 -> EOF
0000B4FF 750A                  <1>       jnz    short csftdf2_rw_f_clust_ok
                               <1>
0000B501 81FB00000800          <1>       cmp    ebx, Cluster_Buffer + 65536
0000B507 72E9                  <1>       jb     short csftdf2_write_df_clust_next
                               <1>
0000B509 EB82                  <1>       jmp    short csftdf2_read_sf_cluster
                               <1>
                               <1> csftdf2_rw_f_clust_ok:
0000B50B 803D[38650100]00      <1>       cmp    byte [csftdf_percentage], 0
0000B512 0F86B2010000          <1>         jna     csftdf2_save_fat_file_4 ; 25/03/2016
                               <1>
                               <1>       ; "100%"
0000B518 BF[3B130100]          <1>       mov    edi, percentagestr
0000B51D B031                  <1>       mov    al, '1'
0000B51F AA                    <1>       stosb
0000B520 B030                  <1>       mov    al, '0'
0000B522 AA                    <1>       stosb
0000B523 AA                    <1>       stosb
                               <1>
0000B524 8A3D[39650100]        <1>       mov    bh, [csftdf_videopage]
0000B52A 668B15[3A650100]      <1>       mov    dx, [csftdf_cursorpos]
0000B531 B402                  <1>       mov    ah, 2
0000B533 E8625FFFFF            <1>       call   _int10h
                               <1>
0000B538 BE[3B130100]          <1>       mov    esi, percentagestr
0000B53D E81BAEFFFF            <1>       call   print_msg
                               <1>
0000B542 E983010000            <1>         jmp     csftdf2_save_fat_file_4
                               <1>
                               <1> csftdf2_load_fs_file:
                               <1>       ; temporary - 18/03/2016
0000B547 E96F020000            <1>         jmp     csftdf2_read_error
                               <1>
                               <1> csftdf2_write_file_sectors:
                               <1>       ; 19/03/2016
0000B54C 807E0300              <1>       cmp    byte [esi+LD_FATType], 0
0000B550 0F86F1050000          <1>         jna     csftdf2_write_fs_file_sectors
                               <1>
                               <1> csftdf2_write_fat_file_sectors:
                               <1>       ; 19/03/2016
                               <1>       ; 18/03/2016
                               <1>       ; return:
                               <1>       ;   CF = 0 & EDX > 0 -> END OF FILE
                               <1>       ;   CF = 0 & EDX = 0 -> not EOF
                               <1>       ;   CF = 1 -> write error (error code in AL)
                               <1>
                               <1> csftdf2_write_fat_file_secs_0:
0000B556 8B15[14650100]        <1>       mov    edx, [csftdf_filesize]
0000B55C 2B15[34650100]        <1>       sub    edx, [csftdf_df_wbytes]
0000B562 3B15[2C650100]        <1>       cmp    edx, [csftdf_w_size]
0000B568 7306                  <1>       jnb    short csftdf2_write_fat_file_secs_1
0000B56A 8915[2C650100]        <1>       mov    [csftdf_w_size], edx
                               <1>
                               <1> csftdf2_write_fat_file_secs_1:
0000B570 A1[2C650100]          <1>       mov    eax, [csftdf_w_size]
0000B575 29D2                  <1>       sub    edx, edx
0000B577 0FB74E11              <1>       movzx  ecx, word [esi+LD_BPB+BytesPerSec]
0000B57B 01C8                  <1>       add    eax, ecx
0000B57D 48                    <1>       dec    eax
0000B57E F7F1                  <1>       div    ecx
0000B580 89C1                  <1>       mov    ecx, eax ; sector count
0000B582 A1[24650100]          <1>       mov    eax, [csftdf_df_cluster]
                               <1>
                               <1>       ; EBX = memory block address (current)
                               <1>
0000B587 E8A20F0000            <1>       call   write_fat_file_sectors
0000B58C 7259                  <1>       jc     short csftdf2_write_fat_file_secs_4
                               <1>
                               <1>       ; EBX = next memory address
                               <1>
0000B58E A1[34650100]          <1>       mov    eax, [csftdf_df_wbytes]
0000B593 0305[2C650100]        <1>       add    eax, [csftdf_w_size]
0000B599 8B15[14650100]        <1>       mov    edx, [csftdf_filesize]
0000B59F 39D0                  <1>       cmp    eax, edx
0000B5A1 7344                  <1>       jnb    short csftdf2_write_fat_file_secs_4
0000B5A3 A3[34650100]          <1>       mov    [csftdf_df_wbytes], eax
                               <1>       ;
0000B5A8 A3[D6640100]          <1>       mov    [DestinationFile_DirEntry+DirEntry_FileSize], eax
                               <1>
0000B5AD 53                    <1>       push   ebx ; *
                               <1>
0000B5AE 803D[12650100]01      <1>       cmp    byte [DestinationFileFound], 1
0000B5B5 7210                  <1>       jb     short csftdf2_write_fat_file_secs_2
                               <1>
                               <1>       ; get next cluster (csftdf_w_size! bytes)
0000B5B7 A1[24650100]          <1>       mov    eax, [csftdf_df_cluster]
0000B5BC E887050000            <1>       call   get_next_cluster
0000B5C1 731C                  <1>       jnc    short csftdf2_write_fat_file_secs_3
                               <1>
0000B5C3 21C0                  <1>       and    eax, eax ; end of cluster chain!?
0000B5C5 7521                  <1>       jnz    short csftdf2_write_fat_file_secs_5 ; disk error !
                               <1>
                               <1> csftdf2_write_fat_file_secs_2:
0000B5C7 A1[24650100]          <1>       mov    eax, [csftdf_df_cluster] ; last cluster
0000B5CC E8800E0000            <1>       call   add_new_cluster
0000B5D1 7215                  <1>       jc     short csftdf2_write_fat_file_secs_5
                               <1>
```

```
                             <1>       ; NOTE: Destination file size may be bigger than
                             <1>       ; source file size when the last reading fails after here.
                             <1>       ; (The last -empty- cluster of destination file must be
                             <1>       ; truncated and LMDT must be current date&time for partial
                             <1>       ; copy result!)
0000B5D3 8B15[2C650100]      <1>       mov    edx, [csftdf_w_size] ; bytes per cluster
0000B5D9 0115[D6640100]      <1>       add    [DestinationFile_DirEntry+DirEntry_FileSize], edx
                             <1>
                             <1> csftdf2_write_fat_file_secs_3:
0000B5DF 5B                  <1>       pop    ebx ; *
0000B5E0 29D2                <1>       sub    edx, edx ; 0
0000B5E2 A3[24650100]        <1>       mov    [csftdf_df_cluster], eax ; next cluster
                             <1>
                             <1> csftdf2_write_fat_file_secs_4:
0000B5E7 C3                  <1>       retn
                             <1>
                             <1> csftdf2_write_fat_file_secs_5:
0000B5E8 5B                  <1>       pop    ebx ; *
                             <1>       ; 16/10/2016 (1Dh -> 18)
0000B5E9 B812000000          <1>       mov    eax, 18 ; Write error !
0000B5EE C3                  <1>       retn
                             <1>
                             <1> csftdf2_save_file:
                             <1>       ; 09/12/2017
                             <1>       ; 25/03/2016
                             <1>       ; 19/03/2016
                             <1>       ; 18/03/2016
0000B5EF 8B35[40650100]      <1>       mov    esi, [csftdf_df_drv_dt] ; logical dos drv desc. tbl.
                             <1>
0000B5F5 8B1D[18650100]      <1>       mov    ebx, [csftdf_sf_mem_addr] ; memory block address
                             <1>
0000B5FB 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
0000B5FF 0F86F4010000        <1>         jna     csftdf2_save_fs_file
                             <1>
                             <1> csftdf2_save_fat_file:
0000B605 53                  <1>       push   ebx; *
                             <1>
0000B606 803D[38650100]00    <1>       cmp    byte [csftdf_percentage], 0
0000B60D 7724                <1>       ja     short csftdf2_save_fat_file_0
                             <1>
                             <1>       ; Set cursor position
                             <1>       ; AH= 02h, BH= Page Number, DH= Row, DL= Column
0000B60F 8A3D[39650100]      <1>       mov    bh, [csftdf_videopage]
0000B615 668B15[3A650100]    <1>       mov    dx, [csftdf_cursorpos]
0000B61C B402                <1>       mov    ah, 2
0000B61E E8775EFFFF          <1>       call   _int10h
                             <1>
0000B623 BE[2F130100]        <1>       mov    esi, msg_writing
0000B628 E830ADFFFF          <1>       call   print_msg
                             <1>
                             <1> csftdf2_save_fat_file_next:
0000B62D 8B35[40650100]      <1>       mov    esi, [csftdf_df_drv_dt] ; 25/03/2016
                             <1>
                             <1> csftdf2_save_fat_file_0:
0000B633 5B                  <1>       pop    ebx ; *
                             <1>
                             <1> csftdf2_save_fat_file_1:
0000B634 E813FFFFFF          <1>       call   csftdf2_write_file_sectors ; 19/03/2016
0000B639 0F827E010000        <1>         jc      csftdf2_rw_error ; eocc! or disk error!
                             <1>
0000B63F 09D2                <1>       or     edx, edx ; edx > 0 -> EOF
0000B641 756D                <1>         jnz     short csftdf2_save_fat_file_3 ; 25/03/2016
                             <1>
0000B643 803D[38650100]00    <1>       cmp    byte [csftdf_percentage], 0
0000B64A 76E8                <1>       jna    short csftdf2_save_fat_file_1
                             <1>
0000B64C B020                <1>       mov    al, 20h
0000B64E BF[3B130100]        <1>       mov    edi, percentagestr
0000B653 AA                  <1>       stosb
0000B654 AA                  <1>       stosb
0000B655 A1[34650100]        <1>       mov    eax, [csftdf_df_wbytes]
0000B65A BA64000000          <1>       mov    edx, 100
0000B65F F7E2                <1>       mul    edx
0000B661 8B0D[14650100]      <1>       mov    ecx, [csftdf_filesize]
0000B667 F7F1                <1>       div    ecx
0000B669 B10A                <1>       mov    cl, 10
0000B66B F6F1                <1>       div    cl
0000B66D 80C430              <1>       add    ah, '0'
0000B670 8827                <1>       mov    [edi], ah
0000B672 20C0                <1>       and    al, al
0000B674 740A                <1>       jz     short csftdf2_save_fat_file_2
0000B676 4F                  <1>       dec    edi
                             <1>       ;cbw
0000B677 30E4                <1>       xor    ah, ah ; 09/12/2017
0000B679 F6F1                <1>       div    cl
0000B67B 80C430              <1>       add    ah, '0'
0000B67E 8827                <1>       mov    [edi], ah
                             <1>       ;and   al, al
                             <1>       ;jz    short csftdf2_save_fat_file_2
                             <1>       ;dec   edi
                             <1>       ;mov   [edi], '1' ; 100%
                             <1>
                             <1> csftdf2_save_fat_file_2:
0000B680 53                  <1>       push   ebx ; *
                             <1>
0000B681 E802000000          <1>       call   csftdf2_print_wr_percentage ; 25/03/2016
                             <1>
0000B686 EBA5                <1>         jmp     csftdf2_save_fat_file_next
                             <1>
                             <1> csftdf2_print_wr_percentage:
                             <1>       ; Set cursor position
                             <1>       ; AH= 02h, BH= Page Number, DH= Row, DL= Column
0000B688 8A3D[39650100]      <1>       mov    bh, [csftdf_videopage]
```

```
0000B68E 668B15[3A650100]    <1>        mov    dx, [csftdf_cursorpos]
0000B695 B402                <1>        mov    ah, 2
0000B697 E8FE5DFFFF          <1>        call   _int10h
                             <1>
0000B69C BE[2F130100]        <1>        mov    esi, msg_writing
0000B6A1 E8B7ACFFFF          <1>        call   print_msg
                             <1>
0000B6A6 BE[3B130100]        <1>        mov    esi, percentagestr
                             <1>        ;call  print_msg
                             <1>        ;retn
0000B6AB E9ADACFFFF          <1>        jmp    print_msg
                             <1>
                             <1> csftdf2_save_fat_file_3:
0000B6B0 803D[38650100]00    <1>        cmp    byte [csftdf_percentage], 0
0000B6B7 7611                <1>          jna     csftdf2_save_fat_file_4 ; 25/03/2016
                             <1>
                             <1>        ; "100%"
0000B6B9 BF[3B130100]        <1>        mov    edi, percentagestr
0000B6BE B031                <1>        mov    al, '1'
0000B6C0 AA                  <1>        stosb
0000B6C1 B030                <1>        mov    al, '0'
0000B6C3 AA                  <1>        stosb
0000B6C4 AA                  <1>        stosb
                             <1>
0000B6C5 E8BEFFFFFF          <1>        call   csftdf2_print_wr_percentage
                             <1>
                             <1> csftdf2_save_fat_file_4:
0000B6CA 803D[12650100]00    <1>        cmp    byte [DestinationFileFound], 0
0000B6D1 7647                <1>        jna    short csftdf2_save_fat_file_6
                             <1>
0000B6D3 8B35[40650100]      <1>        mov    esi, [csftdf_df_drv_dt] ; 31/03/2016
                             <1>
0000B6D9 A1[24650100]        <1>        mov    eax, [csftdf_df_cluster] ; last cluster
0000B6DE E865040000          <1>        call   get_next_cluster
0000B6E3 7235                <1>        jc     short csftdf2_save_fat_file_6 ; eocc! or disk error!
                             <1>
0000B6E5 A1[24650100]        <1>        mov    eax, [csftdf_df_cluster] ; last cluster
                             <1>        ;xor   ecx, ecx
                             <1>        ;mov   [FAT_ClusterCounter], ecx ; 0 ; reset
                             <1>        ;dec   ecx ; 0FFFFFFFFh
                             <1>        ;shr   ecx, 4 ; 28 bit ; 0FFFFFFFh
0000B6EA B9FFFFFF0F          <1>        mov    ecx, 0FFFFFFFh
0000B6EF E87E070000          <1>        call   update_cluster
0000B6F4 7224                <1>        jc     short csftdf2_save_fat_file_6 ; really last cluster!?
                             <1>
0000B6F6 A3[24650100]        <1>        mov    [csftdf_df_cluster], eax ; next cluster
                             <1>
                             <1>        ; byte [FAT_BuffValidData] = 2
0000B6FB E82F0A0000          <1>        call   save_fat_buffer
0000B700 730E                <1>        jnc    short csftdf2_save_fat_file_5
                             <1>
0000B702 8B15[14650100]      <1>        mov    edx, [csftdf_filesize]
0000B708 8915[D6640100]      <1>        mov    [DestinationFile_DirEntry+DirEntry_FileSize], edx
0000B70E EB58                <1>        jmp    short csftdf2_save_fat_file_err3
                             <1>
                             <1> csftdf2_save_fat_file_5:
0000B710 A1[24650100]        <1>        mov    eax, [csftdf_df_cluster]
                             <1>
                             <1>        ; EAX = First cluster to be truncated/unlinked
                             <1>        ; ESI = Logical dos drive description table address
0000B715 E8580C0000          <1>        call   truncate_cluster_chain
                             <1>
                             <1> csftdf2_save_fat_file_6:
                             <1>        ; 28/03/2016
0000B71A BE[45640100]        <1>        mov    esi, SourceFile_DirEntry+DirEntry_Attr ; +11 to + 18
0000B71F BF[C5640100]        <1>        mov    edi, DestinationFile_DirEntry+DirEntry_Attr ; +11 to + 18
0000B724 A4                  <1>        movsb ; +11
0000B725 A5                  <1>        movsd ; +12 .. +15
0000B726 66A5                <1>        movsw ; +16 .. +17
                             <1>               ; + 18
0000B728 83C604              <1>        add    esi, 4
0000B72B 83C704              <1>        add    edi, 4
0000B72E A5                  <1>        movsd  ; DirEntry_WrtTime ; +22 .. +25
                             <1>
0000B72F 8B15[14650100]      <1>        mov    edx, [csftdf_filesize]
0000B735 8915[D6640100]      <1>        mov    [DestinationFile_DirEntry+DirEntry_FileSize], edx
                             <1>
0000B73B E8BAF0FFFF          <1>        call   convert_current_date_time
                             <1>        ; DX = Date in dos dir entry format
                             <1>        ; AX = Time in dos dir entry format
0000B740 EB4D                <1>        jmp    short csftdf2_save_fat_file_7
                             <1>
                             <1> csftdf2_save_fat_file_err1:
0000B742 5B                  <1>        pop    ebx ; *
                             <1> csftdf2_save_fat_file_err2:
0000B743 A1[34650100]        <1>        mov    eax, [csftdf_df_wbytes]
0000B748 8B15[D6640100]      <1>        mov    edx, [DestinationFile_DirEntry+DirEntry_FileSize]
0000B74E 39C2                <1>        cmp    edx, eax
0000B750 7616                <1>        jna    short csftdf2_save_fat_file_err3
0000B752 A1[24650100]        <1>        mov    eax, [csftdf_df_cluster] ; last (empty) cluster
                             <1>        ; ESI = Logical dos drive description table address
0000B757 E8160C0000          <1>        call   truncate_cluster_chain
0000B75C 720A                <1>        jc     short csftdf2_save_fat_file_err3
0000B75E A1[34650100]        <1>        mov    eax, [csftdf_df_wbytes]
0000B763 A3[D6640100]        <1>        mov    [DestinationFile_DirEntry+DirEntry_FileSize], eax
                             <1> csftdf2_save_fat_file_err3:
0000B768 E88DF0FFFF          <1>        call   convert_current_date_time
                             <1>        ; DX = Date in dos dir entry format
                             <1>        ; AX = Time in dos dir entry format
0000B76D C605[C7640100]00    <1>        mov    byte [DestinationFile_DirEntry+DirEntry_CrtTimeTenth], 0
0000B774 66A3[C8640100]      <1>        mov    [DestinationFile_DirEntry+DirEntry_CrtTime], ax
0000B77A 668915[CA640100]    <1>        mov    [DestinationFile_DirEntry+DirEntry_CrtDate], dx
0000B781 66A3[D0640100]      <1>        mov    [DestinationFile_DirEntry+DirEntry_WrtTime], ax
```

```
0000B787 668915[D2640100]    <1>       mov    [DestinationFile_DirEntry+DirEntry_WrtDate], dx
0000B78E F9                  <1>       stc
                             <1> csftdf2_save_fat_file_7:
0000B78F 9C                  <1>       pushf
0000B790 668915[CC640100]    <1>       mov    [DestinationFile_DirEntry+DirEntry_LastAccDate], dx
0000B797 BE[BA640100]        <1>       mov    esi, DestinationFile_DirEntry
0000B79C BF00000800          <1>       mov    edi, Directory_Buffer
0000B7A1 0FB70D[E2640100]    <1>       movzx  ecx, word [DestinationFile_DirEntryNumber] ; (<2048)
0000B7A8 66C1E105            <1>       shl    cx, 5 ; 32 * directory entry number
0000B7AC 01CF                <1>       add    edi, ecx
                             <1>       ;mov   ecx, 8
0000B7AE 66B90800            <1>       mov    cx, 8
0000B7B2 F3A5                <1>       rep    movsd
0000B7B4 9D                  <1>       popf
0000B7B5 730B                <1>       jnc    short csftdf2_write_file_OK
                             <1>
                             <1> csftdf2_write_error:
                             <1>       ; 18/03/2016
0000B7B7 B01D                <1>       mov    al, 1Dh ; write error
0000B7B9 EB02                <1>       jmp    short csftdf2_rw_error
                             <1>
                             <1>       ; 16/03/2016
                             <1> csftdf2_read_error:
0000B7BB B011                <1>       mov    al, 17 ; ; Drive not ready or read error!
                             <1> csftdf2_rw_error:
0000B7BD A2[11650100]        <1>       mov    [csftdf_rw_err], al
                             <1>
                             <1> csftdf2_write_file_OK:
                             <1>       ; 18/03/2016
0000B7C2 C605[28610100]02    <1>       mov    byte [DirBuff_ValidData], 2
0000B7C9 E8CAF0FFFF          <1>       call   save_directory_buffer
                             <1>
                             <1>       ; Update last modification date&time of destination
                             <1>       ; file's (parent) directory
0000B7CE E860F1FFFF          <1>       call   update_parent_dir_lmdt
                             <1>       ;
0000B7D3 A1[18650100]        <1>       mov    eax, [csftdf_sf_mem_addr] ; start address
                             <1>
0000B7D8 21C0                <1>       and    eax, eax
0000B7DA 750E                <1>       jnz    short csftdf2_dealloc_mblock
                             <1>
0000B7DC 88C5                <1>       mov    ch, al ; 0 (Cluster r/w, not full loading)
                             <1> csftdf2_dealloc_retn:
0000B7DE 8A0D[11650100]      <1>       mov    cl, [csftdf_rw_err]
0000B7E4 A1[24650100]        <1>       mov    eax, [csftdf_df_cluster]
0000B7E9 C3                  <1>       retn
                             <1>
                             <1> csftdf2_dealloc_mblock:
0000B7EA 8B0D[1C650100]      <1>       mov    ecx, [csftdf_sf_mem_bsize] ; block size
0000B7F0 E83C9EFFFF          <1>       call   deallocate_memory_block
0000B7F5 B5FF                <1>         mov     ch, 0FFh ; (File was full loaded at memory)
0000B7F7 EBE5                <1>       jmp    short csftdf2_dealloc_retn
                             <1>
                             <1> csftdf2_save_fs_file:
                             <1>       ; 16/10/2016 (1Dh -> 18)
                             <1>       ; temporary - (21/03/2016)
0000B7F9 B812000000          <1>       mov    eax, 18 ; write error
0000B7FE F9                  <1>       stc
0000B7FF C3                  <1>       retn
                             <1>
                             <1> create_file:
                             <1>       ; 16/10/2016
                             <1>       ; 24/03/2016, 31/03/2016
                             <1>       ; 20/03/2016, 21/03/2016, 23/03/2016
                             <1>       ; 19/03/2016 (TRDOS 396 = TRDOS v2.0)
                             <1>       ; 03/09/2011 (FILE.ASM, 'proc_create_file')
                             <1>       ; 09/08/2010
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      EAX = File Size
                             <1>       ;      ESI = ASCIIZ File Name
                             <1>       ;      CL = File Attributes
                             <1>       ;      EBX = FFFFFFFFh -> create empty file
                             <1>       ;                    (only for FAT fs)
                             <1>       ; OUTPUT ->
                             <1>       ;     CF = 0 ->
                             <1>       ;      EAX = New file's first cluster
                             <1>       ;      ESI = Logical Dos Drv Descr. Table Addr.
                             <1>       ;      EBX = offset CreateFile_Size
                             <1>       ;      ECX = Sectors per cluster (<256)
                             <1>       ;      EDX = Directory entry index/number (<65536)
                             <1>       ;     CF = 1 -> error code in AL
                             <1>
                             <1> ;     test   cl, 18h (directory or volume name)
                             <1> ;     jnz    short loc_createfile_access_denied
0000B800 80E107              <1>       and    cl, 07h ; S, H, R
0000B803 880D[60650100]      <1>         mov     [createfile_attrib], cl
                             <1>
0000B809 89D9                <1>       mov    ecx, ebx
0000B80B 89F3                <1>       mov    ebx, esi ; ASCIIZ File Name address
0000B80D 29D2                <1>       sub    edx, edx
0000B80F 8A35[FE580100]      <1>         mov     dh, [Current_Drv]
0000B815 BE00010900          <1>         mov     esi, Logical_DOSDisks
0000B81A 01D6                <1>       add    esi, edx
                             <1>
0000B81C 8815[6B650100]      <1>       mov    [createfile_UpdatePDir], dl ; 0 ; 31/03/2016
                             <1>
                             <1>       ; LD_DiskType = 0 for write protection (read only)
0000B822 807E0101            <1>       cmp    byte [esi+LD_DiskType], 1 ; 0 = Invalid
0000B826 730A                <1>       jnb    short loc_createfile_check_file_sytem
                             <1>       ; 16/10/2016 (TRDOS Error code: 30, disk write protected)
0000B828 B81E000000          <1>       mov    eax, 30 ; 13h, MSDOS err : Disk write-protected
0000B82D 66BA0000            <1>       mov    dx, 0
```

```
                              <1>       ; err retn: EDX = 0, EBX = File name offset
                              <1>       ; ESI -> Dos drive description table address
0000B831 C3                   <1>       retn
                              <1>
                              <1> ;loc_createfile_access_denied:
                              <1> ;     mov    eax, 05h ; access denied (invalid attributes input)
                              <1> ;     stc
                              <1> ;     retn
                              <1>
                              <1> loc_createfile_check_file_sytem:
0000B832 807E0301             <1>       cmp    byte [esi+LD_FATType], 1
0000B836 730A                 <1>       jnb    short loc_createfile_chk_empty_FAT_file_sign1
                              <1>
0000B838 A3[4C650100]         <1>       mov    [createfile_size], eax
                              <1>       ; ESI = Logical Dos Drive Description Table address
                              <1>       ; EBX = ASCIIZ File Name address
0000B83D E9FE020000           <1>       jmp    create_fs_file
                              <1>
                              <1> loc_createfile_chk_empty_FAT_file_sign1:
                              <1>       ; ECX = FFFFFFFFh -> create empty file if drive has FAT fs
0000B842 41                   <1>       inc    ecx
0000B843 7506                 <1>       jnz    short loc_createfile_chk_empty_FAT_file_sign2
0000B845 890D[4C650100]       <1>       mov    [createfile_size], ecx ; 0 ; empty file
                              <1>
                              <1> loc_createfile_chk_empty_FAT_file_sign2:
                              <1>       ; 23/03/2016
0000B84B 668B4E11             <1>       mov    cx, [esi+LD_BPB+BytesPerSec]
0000B84F 66890D[68650100]     <1>       mov    [createfile_BytesPerSec], cx
                              <1>
                              <1>       ; EBX = ASCIIZ File Name address
0000B856 0FB65613             <1>       movzx  edx, byte [esi+LD_BPB+SecPerClust]
0000B85A 8815[61650100]       <1>       mov    [createfile_SecPerClust], dl
0000B860 8B4E74               <1>       mov    ecx, [esi+LD_FreeSectors]
0000B863 39D1                 <1>       cmp    ecx, edx ; byte [createfile_SecPerClust]
0000B865 7306                 <1>       jnb    short loc_create_fat_file
                              <1>
                              <1> loc_createfile_insufficient_disk_space:
0000B867 B827000000           <1>       mov    eax, 27h
                              <1> loc_createfile_gffc_retn:
0000B86C C3                   <1>       retn
                              <1>
                              <1> loc_create_fat_file:
0000B86D 891D[44650100]       <1>       mov    [createfile_Name_Offset], ebx
0000B873 890D[48650100]       <1>       mov    [createfile_FreeSectors], ecx
                              <1>
                              <1> loc_createfile_gffc_1:
0000B879 E821050000           <1>       call   get_first_free_cluster
0000B87E 72EC                 <1>       jc     short loc_createfile_gffc_retn
                              <1>
0000B880 A3[50650100]         <1>       mov    [createfile_FFCluster], eax
                              <1>
                              <1> loc_createfile_locate_ffe_on_directory:
                              <1>       ; Current directory fcluster <> Directory buffer cluster
                              <1>       ; Current directory will be reloaded by
                              <1>       ; 'locate_current_dir_file' procedure
                              <1>       ;
                              <1>       ; ESI = Logical Dos Drv Desc. Table Adress
0000B885 56                   <1>       push   esi ; *
0000B886 31C0                 <1>       xor    eax, eax
                              <1>
0000B888 A3[1E610100]         <1>       mov    dword [FAT_ClusterCounter], eax ; 0
                              <1>       ; 21/03/2016
0000B88D A2[6A650100]         <1>       mov    byte [createfile_wfc], al ; 0
                              <1>
0000B892 89C1                 <1>       mov    ecx, eax
0000B894 6649                 <1>       dec    cx ; FFFFh
                              <1>       ; CX = FFFFh -> find first deleted or free entry
                              <1>       ; ESI would be ASCIIZ filename address if the call
                              <1>       ; would not be for first free or deleted dir entry
0000B896 E8D7E7FFFF           <1>       call   locate_current_dir_file
0000B89B 0F83EE000000         <1>       jnc    loc_createfile_set_ff_dir_entry
0000B8A1 5E                   <1>       pop    esi ; *
                              <1>        ; ESI = Logical DOS Drv. Description Table Address
0000B8A2 83F802               <1>       cmp    eax, 2
0000B8A5 7402                 <1>       je     short loc_createfile_add_new_cluster
                              <1> loc_createfile_locate_file_stc_retn:
0000B8A7 F9                   <1>       stc
0000B8A8 C3                   <1>       retn
                              <1>
                              <1> loc_createfile_add_new_cluster:
0000B8A9 803D[FD580100]02     <1>       cmp    byte [Current_FATType], 2
                              <1>       ;cmp   byte [esi+LD_FATType], 2
0000B8B0 770C                 <1>       ja     short loc_createfile_add_new_cluster_check_fsc
0000B8B2 803D[FC580100]01     <1>       cmp    byte [Current_Dir_Level], 1
                              <1>       ;cmp   byte [esi+LD_CDirLevel], 1
0000B8B9 7303                 <1>       jnb    short loc_createfile_add_new_cluster_check_fsc
                              <1>
                              <1>       ;mov   eax, 12
0000B8BB B00C                 <1>       mov    al, 12 ; No more files
                              <1>
                              <1> loc_createfile_anc_retn:
0000B8BD C3                   <1>       retn
                              <1>
                              <1> loc_createfile_add_new_cluster_check_fsc:
0000B8BE 8B0D[48650100]       <1>       mov    ecx, [createfile_FreeSectors]
0000B8C4 0FB605[61650100]     <1>       movzx  eax, byte [createfile_SecPerClust]
0000B8CB 66D1E0               <1>       shl    ax, 1 ; AX = 2 * AX
0000B8CE 39C1                 <1>       cmp    ecx, eax
0000B8D0 7295                 <1>         jb   short loc_createfile_insufficient_disk_space
                              <1>
                              <1> loc_createfile_add_new_subdir_cluster:
0000B8D2 8B15[2D610100]       <1>       mov    edx, [DirBuff_Cluster]
0000B8D8 8915[54650100]       <1>       mov    [createfile_LastDirCluster], edx
```

```
                             <1>
0000B8DE A1[50650100]        <1>       mov    eax, [createfile_FFCluster]
0000B8E3 E846040000          <1>       call   load_FAT_sub_directory
0000B8E8 72D3                <1>       jc     short loc_createfile_anc_retn
                             <1>
                             <1> pass_createfile_add_new_subdir_cluster:
                             <1>       ;movzx eax, word [esi+LD_BPB+BytesPerSec]
0000B8EA 0FB705[68650100]    <1>       movzx  eax, word [createfile_BytesPerSec] ; 23/03/2016
0000B8F1 F7E1                <1>       mul    ecx ; ecx = directory buffer sector count
0000B8F3 89C1                <1>       mov    ecx, eax
0000B8F5 C1E902              <1>       shr    ecx, 2 ; dword count
0000B8F8 29C0                <1>       sub    eax, eax ; 0
0000B8FA F3AB                <1>       rep    stosd
                             <1>       ;
0000B8FC C605[28610100]02    <1>       mov    byte [DirBuff_ValidData], 2
0000B903 E890EFFFFF          <1>       call   save_directory_buffer
0000B908 72B3                <1>       jc     short loc_createfile_anc_retn
                             <1>
                             <1> loc_createfile_save_added_subdir_cluster:
0000B90A A1[54650100]        <1>       mov    eax, [createfile_LastDirCluster]
0000B90F 8B0D[50650100]      <1>       mov    ecx, [createfile_FFCluster]
0000B915 E858050000          <1>       call   update_cluster
0000B91A 7304                <1>       jnc    short loc_createfile_save_fat_buffer_0
0000B91C 09C0                <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
0000B91E 751A                <1>       jnz    short loc_createfile_save_fat_buffer_stc_retn
                             <1>
                             <1> loc_createfile_save_fat_buffer_0:
0000B920 A1[50650100]        <1>       mov    eax, [createfile_FFCluster]
0000B925 A3[54650100]        <1>       mov    [createfile_LastDirCluster], eax
0000B92A B9FFFFFF0F          <1>       mov    ecx, 0FFFFFFFh ; 28 bit
0000B92F E83E050000          <1>       call   update_cluster
0000B934 7306                <1>       jnc    short loc_createfile_save_fat_buffer_1
0000B936 09C0                <1>       or     eax, eax ; Was it free cluster
0000B938 7402                <1>       jz     short loc_createfile_save_fat_buffer_1
                             <1>
                             <1> loc_createfile_save_fat_buffer_stc_retn:
0000B93A F9                  <1>       stc
                             <1> loc_createfile_save_fat_buffer_retn:
                             <1> loc_createfile_gffc_2_stc_retn:
0000B93B C3                  <1>       retn
                             <1>
                             <1> loc_createfile_save_fat_buffer_1:
                             <1>       ; byte [FAT_BuffValidData] = 2
0000B93C E8EE070000          <1>       call   save_fat_buffer
0000B941 72F8                <1>       jc     short loc_createfile_save_fat_buffer_retn
                             <1>
0000B943 803D[1E610100]01    <1>       cmp    byte [FAT_ClusterCounter], 1
0000B94A 7222                <1>       jb     short loc_createfile_save_fat_buffer_2
                             <1>
                             <1>       ; ESI = Logical DOS Drive Description Table address
0000B94C A1[1E610100]        <1>       mov    eax, [FAT_ClusterCounter]
                             <1>
0000B951 C605[1E610100]00    <1>       mov    byte [FAT_ClusterCounter], 0 ; 21/03/2016
                             <1>
0000B958 66BB01FF            <1>       mov    bx, 0FF01h ; add free clusters
0000B95C E863080000          <1>       call   calculate_fat_freespace
                             <1>
                             <1>       ;inc   eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!
                             <1>       ;jnz   short loc_createfile_save_fat_buffer_2
                             <1>
                             <1>       ; ecx > 0 -> Recalculation is needed
0000B961 09C9                <1>       or     ecx, ecx
0000B963 7409                <1>       jz     short loc_createfile_save_fat_buffer_2
                             <1>
0000B965 66BB00FF            <1>       mov    bx, 0FF00h ; ; recalculate free space
0000B969 E856080000          <1>       call   calculate_fat_freespace
                             <1>
                             <1> loc_createfile_save_fat_buffer_2:
                             <1>       ;call  update_parent_dir_lmdt
                             <1>
                             <1> loc_createfile_gffc_2:
0000B96E E82C040000          <1>       call   get_first_free_cluster
0000B973 72C6                <1>       jc     short loc_createfile_gffc_2_stc_retn
                             <1>
0000B975 A3[50650100]        <1>       mov    [createfile_FFCluster], eax
                             <1>
0000B97A A1[54650100]        <1>       mov    eax, [createfile_LastDirCluster]
                             <1>
0000B97F E8AA030000          <1>       call   load_FAT_sub_directory
0000B984 72B5                <1>       jc     short loc_createfile_gffc_2_stc_retn
                             <1>
0000B986 BF00000800          <1>       mov    edi, Directory_Buffer
                             <1>
0000B98B 6629DB              <1>       sub    bx, bx ; directory entry index/number = 0
                             <1>
0000B98E 56                  <1>       push   esi ; * ; 23/03/2016
                             <1>
                             <1> loc_createfile_set_ff_dir_entry:
0000B98F 66891D[62650100]    <1>       mov    [createfile_DirIndex], bx
                             <1>
                             <1>         ; EDI = Directory entry address
0000B996 8B35[44650100]      <1>       mov    esi, [createfile_Name_Offset]
0000B99C A1[50650100]        <1>       mov    eax, [createfile_FFCluster]
0000B9A1 A3[58650100]        <1>       mov    [createfile_Cluster], eax ; 24/03/2016
0000B9A6 B5FF                <1>       mov    ch, 0FFh
0000B9A8 8A0D[60650100]      <1>         mov cl, [createfile_attrib] ; file attributes
                             <1>       ; CH > 0 -> File size is in [EBX]
0000B9AE BB[4C650100]        <1>       mov    ebx, createfile_size
                             <1>
0000B9B3 E803EEFFFF          <1>       call   make_directory_entry
                             <1>
0000B9B8 5E                  <1>       pop    esi ; * ; ESI = Logical Dos Drv Desc. Table address
                             <1>
```

```
0000B9B9 C605[28610100]02    <1>       mov    byte [DirBuff_ValidData], 2
0000B9C0 E8D3EEFFFF          <1>       call   save_directory_buffer
0000B9C5 7221                <1>       jc     short loc_createfile_set_ff_dir_entry_retn
                             <1>
0000B9C7 C605[6B650100]01    <1>       mov    byte [createfile_UpdatePDir], 1 ; 31/03/2016
                             <1>
                             <1> loc_createfile_get_set_write_file_cluster:
0000B9CE A1[4C650100]        <1>       mov    eax, [createfile_size]
0000B9D3 09C0                <1>       or     eax, eax
0000B9D5 7570                <1>       jnz    short loc_createfile_get_set_wfc_cont
0000B9D7 40                  <1>       inc    eax
                             <1>       ; 23/03/2016
0000B9D8 0FB61D[61650100]    <1>       movzx  ebx, byte [createfile_SecPerClust]
                             <1>       ;movzx ecx, word [esi+LD_BPB+BytesPerSec] ; 512
0000B9DF 0FB70D[68650100]    <1>         movzx   ecx, word [createfile_BytesPerSec] ; 512
0000B9E6 EB7C                <1>       jmp    loc_createfile_set_cluster_count
                             <1>
                             <1> loc_createfile_set_ff_dir_entry_retn:
0000B9E8 C3                  <1>       retn
                             <1>
                             <1> loc_createfile_write_fcluster_to_disk:
0000B9E9 034668              <1>       add    eax, [esi+LD_DATABegin] ; convert to physical address
0000B9EC BB00000700          <1>       mov    ebx, Cluster_Buffer
                             <1>       ; ESI = Logical DOS Drv. Desc. Tbl. address
                             <1>       ; EAX = Disk address
                             <1>       ; EBX = Sector Buffer
                             <1>       ; ECX = sectors per cluster
0000B9F1 E8D33D0000          <1>       call   disk_write
0000B9F6 7211                <1>       jc     short loc_createfile_dsk_wr_err
                             <1>
                             <1> loc_createfile_update_fat_cluster:
                             <1>       ; 21/03/2016
0000B9F8 803D[6A650100]00    <1>       cmp    byte [createfile_wfc], 0
0000B9FF 7712                <1>       ja     short loc_createfile_update_fat_cluster_n1
                             <1>
0000BA01 FE05[6A650100]      <1>       inc    byte [createfile_wfc] ; 1
0000BA07 EB24                <1>       jmp    short loc_createfile_update_fat_cluster_n2
                             <1>
                             <1> loc_createfile_dsk_wr_err:
                             <1>       ; 16/10/2016 (1Dh -> 18)
                             <1>       ; 23/03/2016
0000BA09 B812000000          <1>       mov    eax, 18 ; Drive not ready or write error !
0000BA0E E9BD000000          <1>       jmp    loc_createfile_stc_retn
                             <1>
                             <1> loc_createfile_update_fat_cluster_n1:
0000BA13 A1[5C650100]        <1>       mov    eax, [createfile_PCluster]
0000BA18 8B0D[58650100]      <1>       mov    ecx, [createfile_Cluster]
0000BA1E E84F040000          <1>       call   update_cluster
0000BA23 7308                <1>       jnc    short loc_createfile_update_fat_cluster_n2
0000BA25 09C0                <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
0000BA27 0F85A3000000        <1>       jnz    loc_createfile_stc_retn
                             <1>
                             <1> loc_createfile_update_fat_cluster_n2:
0000BA2D A1[58650100]        <1>         mov  eax, [createfile_Cluster]
0000BA32 B9FFFFFF0F          <1>       mov    ecx, 0FFFFFFFh
0000BA37 E836040000          <1>       call   update_cluster
0000BA3C 734E                <1>       jnc    short loc_createfile_save_fat_buffer_3
0000BA3E 09C0                <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
0000BA40 744A                <1>       jz     short loc_createfile_save_fat_buffer_3
                             <1>
                             <1> loc_createfile_upd_fat_fcluster_stc_retn:
0000BA42 E989000000          <1>       jmp    loc_createfile_stc_retn
                             <1>
                             <1> loc_createfile_get_set_wfc_cont:
                             <1>       ;movzx ecx, word [esi+LD_BPB+BytesPerSec] ; 512
0000BA47 0FB70D[68650100]    <1>       movzx  ecx, word [createfile_BytesPerSec] ; 512
0000BA4E 01C8                <1>       add    eax, ecx
0000BA50 48                  <1>       dec    eax  ; add eax, 511
0000BA51 29D2                <1>       sub    edx, edx
0000BA53 F7F1                <1>       div    ecx
0000BA55 0FB61D[61650100]    <1>       movzx  ebx, byte [createfile_SecPerClust]
0000BA5C 01D8                <1>       add    eax, ebx
0000BA5E 48                  <1>       dec    eax  ; add eax, SecPerClust - 1
0000BA5F 6631D2              <1>       xor    dx, dx
0000BA62 F7F3                <1>       div    ebx
                             <1>
                             <1> loc_createfile_set_cluster_count:
0000BA64 A3[64650100]        <1>       mov    [createfile_CCount], eax
                             <1>
0000BA69 BF00000700          <1>       mov    edi, Cluster_Buffer
0000BA6E 89C8                <1>       mov    eax, ecx ; Bytes per Sector
0000BA70 F7E3                <1>       mul    ebx ; Sectors per Cluster
                             <1>       ; EAX = Bytes per Cluster
0000BA72 89C1                <1>       mov    ecx, eax
0000BA74 C1E902              <1>       shr    ecx, 2 ; dword count
0000BA77 31C0                <1>       xor    eax, eax
0000BA79 F3AB                <1>       rep    stosd ; clear cluster buffer
                             <1>
0000BA7B A1[58650100]        <1>       mov    eax, [createfile_Cluster] ; 24/03/2016
                             <1>
0000BA80 89D9                <1>       mov    ecx, ebx
                             <1>
                             <1> loc_createfile_get_set_wf_fclust_cont:
0000BA82 83E802              <1>       sub    eax, 2
0000BA85 F7E1                <1>       mul    ecx
                             <1>       ; EAX = Logical DOS disk address (offset)
0000BA87 E95DFFFFFF          <1>         jmp     loc_createfile_write_fcluster_to_disk
                             <1>
                             <1> loc_createfile_save_fat_buffer_3:
                             <1>       ; byte [FAT_BuffValidData] = 2
0000BA8C E89E060000          <1>       call   save_fat_buffer
0000BA91 723D                <1>       jc     loc_createfile_stc_retn
                             <1>
                             <1>       ; 21/03/2016
0000BA93 803D[1E610100]01    <1>       cmp    byte [FAT_ClusterCounter], 1
0000BA9A 721B                <1>       jb     short loc_createfile_save_fat_buffer_4
                             <1> 
                             <1>       ; ESI = Logical DOS Drive Description Table address
0000BA9C A1[1E610100]        <1>       mov    eax, [FAT_ClusterCounter]
0000BAA1 66BB01FF            <1>       mov    bx, 0FF01h ; add free clusters
0000BAA5 E81A070000          <1>       call   calculate_fat_freespace
                             <1> 
                             <1>       ;inc   eax ; 0FFFFFFFFh -> 0 ; recalculation is needed!
                             <1>       ;jnz   short loc_createfile_save_fat_buffer_4
                             <1> 
                             <1>       ; ecx > 0 -> Recalculation is needed
0000BAAA 09C9                <1>       or     ecx, ecx
0000BAAC 7409                <1>       jz     short loc_createfile_save_fat_buffer_4
                             <1> 
0000BAAE 66BB00FF            <1>       mov    bx, 0FF00h ; ; recalculate free space
0000BAB2 E80D070000          <1>       call   calculate_fat_freespace
                             <1> 
                             <1> loc_createfile_save_fat_buffer_4:
0000BAB7 FF0D[64650100]      <1>       dec    dword [createfile_CCount]
                             <1>       ;jz    short loc_createfile_upd_dir_modif_date_time
0000BABD 743F                <1>       jz     short loc_createfile_stc_retn_cc ; 31/03/2016
                             <1> 
                             <1> loc_createfile_get_set_write_next_cluster:
0000BABF E8DB020000          <1>       call   get_first_free_cluster
0000BAC4 720A                <1>       jc     short loc_createfile_stc_retn
                             <1> 
                             <1> loc_createfile_get_set_write_next_cluster_1:
0000BAC6 83F8FF              <1>       cmp    eax, 0FFFFFFFFh
0000BAC9 7213                <1>       jb     short loc_createfile_get_set_write_next_cluster_2
                             <1> 
                             <1> loc_createfile_wnc_insufficient_disk_space:
0000BACB B827000000          <1>       mov    eax, 27h ; Insufficient disk space
                             <1> 
                             <1> loc_createfile_stc_retn:
0000BAD0 803D[6A650100]01    <1>       cmp    byte [createfile_wfc], 1
0000BAD7 7324                <1>       jnb    short loc_createfile_err_retn
0000BAD9 C3                  <1>       retn
                             <1> 
                             <1> loc_createfile_wnc_inv_format_retn:
                             <1>       ;mov   eax, 28
0000BADA B01C                <1>       mov    al, 28 ; Invalid format
0000BADC EBF2                <1>       jmp    short loc_createfile_stc_retn
                             <1> 
                             <1> loc_createfile_get_set_write_next_cluster_2:
0000BADE 83F802              <1>       cmp    eax, 2
0000BAE1 72F7                <1>       jb     short loc_createfile_wnc_inv_format_retn
                             <1> 
                             <1> loc_createfile_get_set_write_next_cluster_3:
0000BAE3 8B0D[58650100]      <1>       mov    ecx, [createfile_Cluster]
0000BAE9 A3[58650100]        <1>       mov    [createfile_Cluster], eax
0000BAEE 890D[5C650100]      <1>       mov    [createfile_PCluster], ecx
0000BAF4 0FB60D[61650100]    <1>       movzx  ecx, byte [createfile_SecPerClust]
0000BAFB EB85                <1>       jmp    short loc_createfile_get_set_wf_fclust_cont
                             <1> 
                             <1> loc_createfile_err_retn:
0000BAFD F9                  <1>       stc
                             <1> 
                             <1> ;loc_createfile_upd_dir_modif_date_time:
                             <1> loc_createfile_stc_retn_cc: ; 31/03/2016
0000BAFE 9C                  <1>       pushf  ; cpu is here for an error return or completion
0000BAFF 50                  <1>       push   eax ; error code if cf = 1
                             <1> 
                             <1>       ;call  update_parent_dir_lmdt
                             <1> 
                             <1> ;loc_createfile_stc_retn_cc:
0000BB00 A1[1E610100]        <1>       mov    eax, [FAT_ClusterCounter]
0000BB05 09C0                <1>       or     eax, eax
0000BB07 741A                <1>       jz     short loc_createfile_stc_retn_pop_eax
0000BB09 8A3D[FE580100]      <1>       mov    bh, [Current_Drv]
0000BB0F B301                <1>       mov    bl, 01h ; BL = 1 -> add clusters
                             <1>       ; NOTE: EAX value will be added to Free Cluster Count
                             <1>       ; (If EAX value is negative, Free Cluster Count will be decreased)
0000BB11 E8AE060000          <1>       call   calculate_fat_freespace
                             <1>         ; ESI = Logical DOS Drive Description Table Address
                             <1>         ;jc  short loc_createfile_stc_retn_pop_eax_cf
0000BB16 21C9                <1>       and    ecx, ecx ; cx = 0 -> valid free sector count
0000BB18 7409                <1>       jz     short loc_createfile_stc_retn_pop_eax
                             <1> 
                             <1> loc_createfile_stc_retn_recalc_FAT_freespace:
0000BB1A 66BB00FF            <1>       mov    bx, 0FF00h ; bh = 0FFh ->
                             <1>       ; ESI = Logical DOS Drv DT Addr
                             <1>       ; BL = 0 -> Recalculate
0000BB1E E8A1060000          <1>       call   calculate_fat_freespace
                             <1> 
                             <1> loc_createfile_stc_retn_pop_eax:
0000BB23 58                  <1>       pop    eax
0000BB24 9D                  <1>       popf
0000BB25 7218                <1>       jc     short loc_createfile_retn
                             <1> 
                             <1> loc_createfile_retn_fcluster:
0000BB27 A1[50650100]        <1>       mov    eax, [createfile_FFCluster]
0000BB2C BB[4C650100]        <1>       mov    ebx, createfile_size
                             <1>       ;movzx ecx, byte [esi+LD_BPB+SecPerClust]
0000BB31 0FB60D[61650100]    <1>       movzx  ecx, byte [createfile_SecPerClust] ; 23/03/2016
0000BB38 0FB715[62650100]    <1>       movzx  edx, word [createfile_DirIndex]
                             <1> 
                             <1> loc_createfile_retn:
0000BB3F C3                  <1>       retn
                             <1> 
                             <1> create_fs_file:
                             <1>       ; temporary (21/03/2016)
```

```
0000BB40 C3                  <1>       retn
                             <1> 
                             <1> delete_fs_file:
                             <1>       ; temporary (28/02/2016)
0000BB41 C3                  <1>       retn
                             <1> 
                             <1> rename_fs_file_or_directory:
0000BB42 C3                  <1>       retn
                             <1> 
                             <1> make_fs_directory:
                             <1>       ; temporary (21/02/2016)
0000BB43 C3                  <1>       retn
                             <1> 
                             <1> add_new_fs_section:
                             <1>       ; temporary (11/03/2016)
0000BB44 C3                  <1>       retn
                             <1> 
                             <1> delete_fs_directory_entry:
                             <1>       ; temporary (11/03/2016)
0000BB45 C3                  <1>       retn
                             <1> 
                             <1> csftdf2_read_fs_file_sectors:
                             <1>       ; temporary (19/03/2016)
0000BB46 C3                  <1>       retn
                             <1> 
                             <1> csftdf2_write_fs_file_sectors:
                             <1>       ; temporary (19/03/2016)
0000BB47 C3                  <1>       retn
                                 %include 'trdosk5.s' ; 24/01/2016
                             <1> ; ****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - File System Procedures : trdosk5s
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 23/10/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 24/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
                             <1> ; DRV_FAT.ASM (21/08/2011)
                             <1> ; ****************************************************************************
                             <1> ; DRV_FAT.ASM (c) 2005-2011 Erdogan TAN [ 07/07/2009 ] Last Update: 21/08/2011
                             <1> 
                             <1> get_next_cluster:
                             <1>       ; 15/10/2016
                             <1>       ; 23/03/2016
                             <1>       ; 01/02/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>       ; 05/07/2011
                             <1>       ; 07/07/2009
                             <1>       ; 2005
                             <1>       ; INPUT ->
                             <1>       ;      EAX = Cluster Number (32 bit)
                             <1>       ;      ESI = Logical DOS Drive Parameters Table
                             <1>       ; OUTPUT ->
                             <1>       ;      cf = 0 -> No Error, EAX valid
                             <1>       ;      cf = 1 & EAX = 0 -> End Of Cluster Chain
                             <1>       ;      cf = 1 & EAX > 0 -> Error
                             <1>       ;      ECX = Current/Previous cluster (if CF = 0)
                             <1>       ;      EAX = Next Cluster Number (32 bit)
                             <1>       ;
                             <1>       ; (Modified registers: EAX, ECX, EBX, EDX)
                             <1> 
0000BB48 A3[12610100]        <1>       mov    [FAT_CurrentCluster], eax
                             <1> check_next_cluster_fat_type:
0000BB4D 29D2                <1>       sub    edx, edx ; 0
0000BB4F 807E0302            <1>         cmp     byte [esi+LD_FATType], 2
0000BB53 7250                <1>       jb     short get_FAT12_next_cluster
0000BB55 0F87AF000000        <1>         ja      get_FAT32_next_cluster
                             <1> get_FAT16_next_cluster:
0000BB5B BB00030000          <1>       mov    ebx, 300h ;768
0000BB60 F7F3                <1>       div    ebx
                             <1>       ; EAX = Count of 3 FAT sectors
                             <1>       ; EDX = Cluster Offset (< 768)
0000BB62 66D1E2              <1>       shl    dx, 1 ; Multiply by 2
0000BB65 89D3                <1>       mov    ebx, edx ; Byte Offset
0000BB67 81C3001C0900        <1>       add    ebx, FAT_Buffer
0000BB6D 66BA0300            <1>       mov    dx, 3
0000BB71 F7E2                <1>       mul    edx
                             <1>       ; EAX = FAT Sector (<= 256)
                             <1>       ; EDX = 0
0000BB73 8A0E                <1>       mov    cl, [esi+LD_Name]
0000BB75 803D[16610100]00    <1>       cmp    byte [FAT_BuffValidData], 0
0000BB7C 0F86CC000000        <1>         jna     load_FAT_sectors0
0000BB82 3A0D[17610100]      <1>       cmp    cl, [FAT_BuffDrvName]
0000BB88 0F85C0000000        <1>         jne     load_FAT_sectors0
0000BB8E 3B05[1A610100]      <1>       cmp    eax, [FAT_BuffSector]
0000BB94 0F85BA000000        <1>         jne     load_FAT_sectors1
                             <1>       ;movzx eax, word [ebx]
0000BB9A 668B03              <1>       mov    ax, [ebx]
                             <1>       ; 01/02/2016
                             <1>       ; DRV_FAT.ASM (21/08/2011) had a FATal bug here !
                             <1>       ; (cmp ah, 0Fh) ! (ax >= FF7h)
                             <1>       ; (how can i do a such mistake!?)
                             <1>       ;cmp   al, 0F7h
                             <1>       ;jb    short loc_pass_gnc_FAT16_eoc_check
                             <1>       ;cmp   ah, 0FFh
                             <1>       ;jb    short loc_pass_gnc_FAT16_eoc_check
0000BB9D 6683F8F7            <1>       cmp    ax, 0FFF7h
0000BBA1 725A                <1>       jb     short loc_pass_gnc_FAT16_eoc_check
                             <1>       ; ax >= FFF7h (cluster 0002h to FFF6h is valid, in use)
0000BBA3 EB56                <1>       jmp    short loc_pass_gnc_FAT16_eoc_check_xor_eax
                             <1> 
                             <1> get_FAT12_next_cluster:
```

```
0000BBA5 BB00040000          <1>       mov    ebx, 400h ;1024
0000BBAA F7F3                <1>       div    ebx
                             <1>       ; EAX = Count of 3 FAT sectors
                             <1>       ; EDX = Cluster Offset (< 1024)
0000BBAC 6650                <1>       push   ax
0000BBAE 66B80300            <1>       mov    ax, 3
0000BBB2 66F7E2              <1>       mul    dx     ; Multiply by 3
0000BBB5 66D1E8              <1>       shr    ax, 1  ; Divide by 2
0000BBB8 6689C3              <1>         mov  bx, ax        ; Byte Offset
0000BBBB 81C3001C0900        <1>       add    ebx, FAT_Buffer
0000BBC1 6658                <1>       pop    ax
0000BBC3 66BA0300            <1>       mov    dx, 3
0000BBC7 F7E2                <1>       mul    edx
                             <1>       ; EAX = FAT Sector (<= 12)
                             <1>       ; EDX = 0
0000BBC9 8A0E                <1>       mov    cl, [esi+LD_Name]
0000BBCB 803D[16610100]00    <1>       cmp    byte [FAT_BuffValidData], 0
0000BBD2 767A                <1>       jna    short load_FAT_sectors0
0000BBD4 3A0D[17610100]      <1>       cmp    cl, [FAT_BuffDrvName]
0000BBDA 7572                <1>       jne    short load_FAT_sectors0
0000BBDC 3B05[1A610100]      <1>       cmp    eax, [FAT_BuffSector]
0000BBE2 7570                <1>       jne    short load_FAT_sectors1
0000BBE4 A1[12610100]        <1>       mov    eax, [FAT_CurrentCluster]
0000BBE9 66D1E8              <1>       shr    ax, 1
                             <1>       ;movzx eax, word [ebx]
0000BBEC 668B03              <1>       mov    ax, [ebx]
0000BBEF 7314                <1>       jnc    short get_FAT12_nc_even
0000BBF1 66C1E804            <1>       shr    ax, 4
                             <1> loc_gnc_fat12_eoc_check:
                             <1>       ;cmp   al, 0F7h
                             <1>       ;jb    short loc_pass_gnc_FAT16_eoc_check
                             <1>       ;cmp   ah, 0Fh
                             <1>       ;jb    short loc_pass_gnc_FAT16_eoc_check
0000BBF5 663DF70F            <1>       cmp    ax, 0FF7h
0000BBF9 7202                <1>       jb     short loc_pass_gnc_FAT16_eoc_check
                             <1>       ; ax >= FF7h (cluster 0002h to FF6h is valid, in use)
                             <1>
                             <1> loc_pass_gnc_FAT16_eoc_check_xor_eax:
0000BBFB 31C0                <1>       xor    eax, eax ; 0
                             <1> loc_pass_gnc_FAT16_eoc_check:
                             <1> loc_pass_gnc_FAT32_eoc_check:
0000BBFD 8B0D[12610100]      <1>       mov    ecx, [FAT_CurrentCluster]
0000BC03 F5                  <1>       cmc
0000BC04 C3                  <1>       retn
                             <1>
                             <1> get_FAT12_nc_even:
0000BC05 80E40F              <1>       and    ah, 0Fh
0000BC08 EBEB                <1>       jmp    short loc_gnc_fat12_eoc_check
                             <1>
                             <1> get_FAT32_next_cluster:
0000BC0A BB80010000          <1>       mov    ebx, 180h ;384
0000BC0F F7F3                <1>       div    ebx
                             <1>       ; EAX = Count of 3 FAT sectors
                             <1>       ; EDX = Cluster Offset (< 384)
0000BC11 66C1E202            <1>       shl    dx, 2  ; Multiply by 4
0000BC15 89D3                <1>       mov    ebx, edx ; Byte Offset
0000BC17 81C3001C0900        <1>       add    ebx, FAT_Buffer
0000BC1D 66BA0300            <1>       mov    dx, 3
0000BC21 F7E2                <1>       mul    edx
                             <1>         ; EAX = FAT Sector (<= 2097152) ; (FFFFFF7h * 4) / 512
                             <1>       ;      for 32KB cluster size:
                             <1>       ;      EAX <= 1024 = (4GB / 32KB) * 4) / 512
                             <1>       ; EDX = 0
0000BC23 8A0E                <1>       mov    cl, [esi+LD_Name]
0000BC25 803D[16610100]00    <1>       cmp    byte [FAT_BuffValidData], 0
0000BC2C 7620                <1>       jna    short load_FAT_sectors0
0000BC2E 3A0D[17610100]      <1>       cmp    cl, [FAT_BuffDrvName]
0000BC34 7518                <1>       jne    short load_FAT_sectors0
0000BC36 3B05[1A610100]      <1>       cmp    eax, [FAT_BuffSector] ; 0, 3, 6, 9 ...
0000BC3C 7516                <1>       jne    short load_FAT_sectors1
0000BC3E 8B03                <1>       mov    eax, [ebx]
0000BC40 25FFFFFF0F          <1>       and    eax, 0FFFFFFFh ; 28 bit Cluster
0000BC45 3DF7FFFF0F          <1>       cmp    eax, 0FFFFFF7h
0000BC4A 72B1                <1>       jb     short loc_pass_gnc_FAT32_eoc_check
                             <1>       ; eax >= FFFFFF7h (cluster 0002h to FFFFFF6h is valid)
0000BC4C EBAD                <1>       jmp    short loc_pass_gnc_FAT16_eoc_check_xor_eax
                             <1>
                             <1> load_FAT_sectors0:
0000BC4E 880D[17610100]      <1>       mov    [FAT_BuffDrvName], cl
                             <1> load_FAT_sectors1:
0000BC54 A3[1A610100]        <1>       mov    [FAT_BuffSector], eax
0000BC59 89C3                <1>       mov    ebx, eax
0000BC5B 034660              <1>         add     eax, [esi+LD_FATBegin]
0000BC5E 807E0302            <1>       cmp    byte [esi+LD_FATType], 2
0000BC62 7706                <1>         ja      short load_FAT_sectors3
0000BC64 0FB74E1C            <1>       movzx  ecx, word [esi+LD_BPB+BPB_FATSz16]
0000BC68 EB03                <1>       jmp    short load_FAT_sectors4
                             <1> load_FAT_sectors3:
0000BC6A 8B4E2A              <1>       mov    ecx, [esi+LD_BPB+BPB_FATSz32]
                             <1> load_FAT_sectors4:
0000BC6D 29D9                <1>       sub    ecx, ebx ; [FAT_BuffSector]
0000BC6F 83F903              <1>         cmp     ecx, 3
0000BC72 7605                <1>         jna     short load_FAT_sectors5
0000BC74 B903000000          <1>       mov    ecx, 3
                             <1> load_FAT_sectors5:
0000BC79 BB001C0900          <1>       mov    ebx, FAT_Buffer
0000BC7E E8553B0000          <1>       call   disk_read
0000BC83 730D                <1>       jnc    short load_FAT_sectors_ok
                             <1>       ; 15/10/2016 (15h -> 17)
                             <1>       ; 23/03/2016 (15h)
0000BC85 B811000000          <1>       mov    eax, 17 ; Drive not ready or read error
0000BC8A C605[16610100]00    <1>       mov    byte [FAT_BuffValidData], 0
0000BC91 C3                  <1>       retn
```

```
                              <1> load_FAT_sectors_ok:
0000BC92 C605[16610100]01     <1>       mov    byte [FAT_BuffValidData], 1
0000BC99 A1[12610100]         <1>       mov    eax, [FAT_CurrentCluster]
0000BC9E E9AAFEFFFF           <1>         jmp     check_next_cluster_fat_type
                              <1> 
                              <1> load_FAT_root_directory:
                              <1>       ; 23/10/2016
                              <1>       ; 15/10/2016
                              <1>       ; 07/02/2016
                              <1>       ; 02/02/2016
                              <1>       ; 01/02/2016 (TRDOS 386 =  TRDOS v2.0)
                              <1>       ; 21/05/2011
                              <1>       ; 22/08/2009
                              <1>       ;
                              <1>       ; INPUT ->
                              <1>       ;      ESI = Logical DOS Drive Description Table
                              <1>       ; OUTPUT ->
                              <1>       ;      cf = 1 -> Root directory could not be loaded
                              <1>       ;          EAX > 0 -> Error number
                              <1>       ;      cf = 0 -> EAX = 0
                              <1>       ;      ECX = Directory buffer size in sectors (CL)
                              <1>       ;      EBX = Directory buffer address
                              <1>       ;      NOTE: DirBuffer_Size is in bytes ! (word)
                              <1>       ;
                              <1>       ; (Modified registers: EAX, ECX, EBX, EDX)
                              <1> 
                              <1>       ; NOTE: Only for FAT12 and FAT16 file systems !
                              <1>       ; (FAT32 fs root dir must be loaded as sub directory)
                              <1> 
0000BCA3 8A1E                 <1>       mov    bl, [esi+LD_Name]
0000BCA5 8A7E03               <1>       mov    bh, [esi+LD_FATType]
                              <1> 
                              <1>       ;mov   [DirBuff_DRV], bl
                              <1>       ;mov   [DirBuff_FATType], bh
0000BCA8 66891D[26610100]     <1>       mov    [DirBuff_DRV], bx
                              <1> 
                              <1>       ;cmp   bh, 2
                              <1>       ;ja    short load_FAT32_root_dir0 ; FAT32 root dir
                              <1> 
                              <1> load_FAT_root_dir0: ; 23/10/2016
0000BCAF 0FB75617             <1>       movzx  edx, word [esi+LD_BPB+RootDirEnts]
                              <1> 
                              <1>       ;or    dx, dx ; 0 for FAT32 file systems
                              <1>       ;jz    short load_FAT32_root_dir0 ; FAT32 root dir
                              <1> 
0000BCB3 6681FA0002           <1>       cmp    dx, 512 ; Number of Root Dir Entries
0000BCB8 7414                 <1>       je     short lrd_mov_ecx_32
0000BCBA 89D0                 <1>       mov    eax, edx
                              <1>       ; 23/10/2016
0000BCBC 89C1                 <1>       mov    ecx, eax
0000BCBE 6683C10F             <1>       add    cx, 15 ; round up
0000BCC2 66C1E904             <1>       shr    cx, 4  ; 16 entries per sector (512/32)
                              <1>       ; ecx = Root directory size in sectors
0000BCC6 66C1E005             <1>       shl    ax, 5 ; Root directory size in bytes
0000BCCA 664A                 <1>       dec    dx    ; Last entry number of root dir
                              <1>       ; cx = Dir Buffer sector count
0000BCCC EB0B                 <1>       jmp    short lrd_check_dir_buffer
                              <1> 
                              <1> lrd_mov_ecx_32:
0000BCCE B920000000           <1>       mov    ecx, 32
0000BCD3 664A                 <1>       dec    dx ; 511
0000BCD5 66B80040             <1>       mov    ax, 32*512
                              <1> 
                              <1> lrd_check_dir_buffer:
0000BCD9 29DB                 <1>       sub    ebx, ebx ; 0
0000BCDB 881D[28610100]       <1>       mov    [DirBuff_ValidData], bl ; 0
0000BCE1 668915[2B610100]     <1>       mov    [DirBuff_LastEntry], dx
0000BCE8 891D[2D610100]       <1>       mov    [DirBuff_Cluster], ebx ; 0
0000BCEE 66A3[31610100]       <1>       mov    [DirBuffer_Size], ax
                              <1> 
0000BCF4 8B4664               <1>       mov    eax, [esi+LD_ROOTBegin]
                              <1> read_directory:
0000BCF7 BB00000800           <1>       mov    ebx, Directory_Buffer
0000BCFC 51                   <1>       push   ecx ; Directory buffer sector count
0000BCFD 53                   <1>       push   ebx
0000BCFE E8D53A0000           <1>       call   disk_read
0000BD03 5B                   <1>       pop    ebx
0000BD04 720B                 <1>       jc     short load_DirBuff_error
                              <1> 
                              <1> validate_DirBuff_and_return:
0000BD06 59                   <1>       pop    ecx ; Number of loaded sectors
0000BD07 C605[28610100]01     <1>       mov    byte [DirBuff_ValidData], 1
0000BD0E 31C0                 <1>       xor    eax, eax ; 0 = no error
0000BD10 C3                   <1>       retn
                              <1> 
                              <1> load_DirBuff_error:
0000BD11 89C8                 <1>       mov    eax, ecx ; remaining sectors
0000BD13 59                   <1>       pop    ecx ; sector count
0000BD14 29C1                 <1>       sub    ecx, eax ; Number of loaded sectors
                              <1>       ; 15/10/2016 (15h -> 17)
0000BD16 B811000000           <1>       mov    eax, 17 ; DRV NOT READY OR READ ERROR !
0000BD1B F9                   <1>       stc
0000BD1C C3                   <1>         retn
                              <1> 
                              <1> load_FAT32_root_directory:
                              <1>       ; 02/02/2016 (TRDOS 386 =  TRDOS v2.0)
                              <1>       ;
                              <1>       ; INPUT ->
                              <1>       ;      ESI = Logical DOS Drive Description Table
                              <1>       ; OUTPUT ->
                              <1>       ;      cf = 1 -> Root directory could not be loaded
                              <1>       ;          EAX > 0 -> Error number
                              <1>       ;      cf = 0 -> EAX = 0
```

```
                               <1>       ;      ECX = Directory buffer size in sectors (CL)
                               <1>       ;      EBX = Directory buffer address
                               <1>       ;      NOTE: DirBuffer_Size is in bytes ! (word)
                               <1>       ;
                               <1>       ; (Modified registers: EAX, ECX, EBX, EDX)
                               <1>
                               <1>
0000BD1D 8A1E                  <1>       mov    bl, [esi+LD_Name]
0000BD1F 8A7E03                <1>       mov    bh, [esi+LD_FATType]
                               <1>
                               <1>       ;mov   [DirBuff_DRV], bl
                               <1>       ;mov   [DirBuff_FATType], bh
0000BD22 66891D[26610100]      <1>       mov    [DirBuff_DRV], bx
                               <1>
                               <1> load_FAT32_root_dir0:
0000BD29 8B4632                <1>       mov    eax, [esi+LD_BPB+FAT32_RootFClust]
0000BD2C EB0C                  <1>       jmp    short load_FAT_sub_dir0
                               <1>
                               <1> load_FAT_sub_directory:
                               <1>       ; 01/02/2016 (TRDOS 386 =  TRDOS v2.0)
                               <1>       ; 05/07/2011
                               <1>       ; 23/08/2009
                               <1>       ;
                               <1>       ; INPUT ->
                               <1>       ;      ESI = Logical DOS Drive Description Table
                               <1>       ;      EAX = Cluster Number
                               <1>       ; OUTPUT ->
                               <1>       ;      cf = 1 -> Sub directory could not be loaded
                               <1>       ;          EAX > 0 -> Error number
                               <1>       ;      cf = 0 -> EAX = 0
                               <1>       ;      ECX = Directory buffer size in sectors (CL)
                               <1>       ;      EBX = Directory buffer address
                               <1>       ;
                               <1>       ;      NOTE: DirBuffer_Size is in bytes ! (word)
                               <1>       ;
                               <1>       ; (Modified registers: EAX, ECX, EBX, EDX)
                               <1>
0000BD2E 8A1E                  <1>       mov    bl, [esi+LD_Name]
0000BD30 8A7E03                <1>       mov    bh, [esi+LD_FATType]
                               <1>
                               <1>       ;mov   [DirBuff_DRV], bl
                               <1>       ;mov   [DirBuff_FATType], bh
0000BD33 66891D[26610100]      <1>       mov    [DirBuff_DRV], bx
                               <1>
                               <1> load_FAT_sub_dir0:
0000BD3A 0FB64E13              <1>       movzx  ecx, byte [esi+LD_BPB+SecPerClust]
                               <1>
0000BD3E 882D[28610100]        <1>       mov    [DirBuff_ValidData], ch ; 0
0000BD44 A3[2D610100]          <1>       mov    [DirBuff_Cluster], eax
                               <1>
0000BD49 0FB74611              <1>       movzx  eax, word [esi+LD_BPB+BytesPerSec]
0000BD4D F7E1                  <1>       mul    ecx
0000BD4F C1E805                <1>       shr    eax, 5 ; directory entry count (dir size / 32)
0000BD52 6648                  <1>       dec    ax ; last entry
0000BD54 66A3[2B610100]        <1>       mov    [DirBuff_LastEntry], ax
                               <1>
0000BD5A A1[2D610100]          <1>       mov    eax, [DirBuff_Cluster]
0000BD5F 83E802                <1>       sub    eax, 2
0000BD62 F7E1                  <1>       mul    ecx
0000BD64 034668                <1>       add    eax, [esi+LD_DATABegin]
                               <1>       ; ecx = sector per cluster (dir buffer size = 32 sectors)
0000BD67 EB8E                  <1>       jmp    short read_directory
                               <1>
                               <1> ; DRV_FS.ASM
                               <1>
                               <1> load_current_FS_directory:
0000BD69 C3                    <1>       retn
                               <1> load_FS_root_directory:
0000BD6A C3                    <1>       retn
                               <1> load_FS_sub_directory:
0000BD6B C3                    <1>       retn
                               <1>
                               <1> read_cluster:
                               <1>       ; 15/10/2016
                               <1>       ; 18/03/2016
                               <1>       ; 16/03/2016
                               <1>       ; 17/02/2016
                               <1>       ; 15/02/2016 (TRDOS 386 =  TRDOS v2.0)
                               <1>       ;
                               <1>       ; INPUT ->
                               <1>       ;      EAX = Cluster Number (Sector index for SINGLIX FS)
                               <1>       ;      ESI = Logical DOS Drive Description Table address
                               <1>       ;      EBX = Cluster (File R/W) Buffer address (max. 64KB)
                               <1>       ;      Only for SINGLIX FS:
                               <1>       ;      EDX = File Number (The 1st FDT address)
                               <1>       ; OUTPUT ->
                               <1>       ;      cf = 1 -> Cluster can not be loaded at the buffer
                               <1>       ;          EAX > 0 -> Error number
                               <1>       ;      cf = 0 -> Cluster has been loaded at the buffer
                               <1>       ;
                               <1>       ; (Modified registers: EAX, ECX, EBX, EDX)
                               <1>
0000BD6C 0FB64E13              <1>       movzx  ecx, byte [esi+LD_BPB+BPB_SecPerClust]
                               <1>       ; CL = 1 = [esi+LD_FS_Reserved2] ; SectPerClust for Singlix FS
                               <1>
                               <1> read_file_sectors: ; 16/03/2016
0000BD70 807E0300              <1>       cmp    byte [esi+LD_FATType], 0
0000BD74 761C                  <1>       jna    short read_fs_cluster
                               <1>
                               <1> read_fat_file_sectors: ; 18/03/2016
0000BD76 83E802                <1>       sub    eax, 2 ; Beginning cluster number is always 2
0000BD79 0FB65613              <1>       movzx  edx, byte [esi+LD_BPB+BPB_SecPerClust] ; 18/03/2016
0000BD7D F7E2                  <1>       mul    edx
```

```
0000BD7F 034668              <1>       add    eax, [esi+LD_DATABegin] ; absolute address of the cluster
                             <1>
                             <1>       ; EAX = Disk sector address
                             <1>       ; ECX = Sector count
                             <1>       ; EBX = Buffer address
                             <1>       ; (EDX = 0)
                             <1>       ; ESI = Logical DOS drive description table address
                             <1>
0000BD82 E8513A0000          <1>       call   disk_read
0000BD87 7306                <1>       jnc    short rclust_retn
                             <1>
                             <1>       ; 15/10/2016 (15h -> 17)
0000BD89 B811000000          <1>       mov    eax, 17 ; Drive not ready or read error !
0000BD8E C3                  <1>       retn
                             <1>
                             <1> rclust_retn:
0000BD8F 29C0                <1>       sub    eax, eax ; 0
0000BD91 C3                  <1>       retn
                             <1>
                             <1> read_fs_cluster:
                             <1>       ; 15/02/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>       ; Singlix FS
                             <1>
                             <1>       ; EAX = Cluster number is sector index number of the file (eax)
                             <1>
                             <1>       ; EDX = File number is the first File Descriptor Table address
                             <1>       ;      of the file. (Absolute address of the FDT).
                             <1>
                             <1>       ; eax = sector index (0 for the first sector)
                             <1>       ; edx = FDT0 address
                             <1>              ; 64 KB buffer = 128 sectors (limit)
0000BD92 B980000000          <1>       mov    ecx, 128 ; maximum count of sectors (before eof)
0000BD97 E801000000          <1>       call   read_fs_sectors
0000BD9C C3                  <1>       retn
                             <1>
                             <1> read_fs_sectors:
                             <1>       ; 15/02/2016 (TRDOS 386 =  TRDOS v2.0)
0000BD9D F9                  <1>       stc
0000BD9E C3                  <1>       retn
                             <1>
                             <1> get_first_free_cluster:
                             <1>       ; 02/03/2016
                             <1>       ; 21/02/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>       ; 26/10/2010 (DRV_FAT.ASM, 'proc_get_first_free_cluster')
                             <1>       ; 10/07/2010
                             <1>       ; INPUT ->
                             <1>       ;      ESI = Logical DOS Drive Description Table address
                             <1>       ; OUTPUT ->
                             <1>       ;      cf = 1 -> Error code in AL (EAX)
                             <1>       ;      cf = 0 ->
                             <1>       ;        EAX = Cluster number
                             <1>       ;        If EAX = FFFFFFFFh -> no free space
                             <1>       ;      If the drive has FAT32 fs:
                             <1>       ;        EBX = FAT32 FSI sector buffer address (if > 0)
                             <1>
0000BD9F 8B4678              <1>       mov    eax, [esi+LD_Clusters]
0000BDA2 40                  <1>       inc    eax ; add eax, 1
0000BDA3 A3[B0630100]        <1>       mov    [gffc_last_free_cluster], eax
                             <1>
0000BDA8 31DB                <1>       xor    ebx, ebx ; 0 ; 02/03/2016
                             <1>
0000BDAA 807E0302            <1>       cmp    byte [esi+LD_FATType], 2
0000BDAE 760E                <1>       jna    short loc_gffc_get_first_fat_free_cluster0
                             <1>
                             <1> loc_gffc_get_first_fat32_free_cluster:
                             <1>       ; 02/03/2016
0000BDB0 E844060000          <1>       call   get_fat32_fsinfo_sector_parms
0000BDB5 7207                <1>       jc     short loc_gffc_get_first_fat_free_cluster0
                             <1>
                             <1> loc_gffc_check_fsinfo_parms:
                             <1>       ;;mov  ebx, DOSBootSectorBuff
                             <1>       ;cmp   dword [ebx], 41615252h
                             <1>       ;jne   short loc_gffc_fat32_fsinfo_err
                             <1>       ;cmp   dword [ebx+484], 61417272h
                             <1>       ;jne   short loc_gffc_fat32_fsinfo_err
                             <1>       ;mov   eax, [ebx+492] ; FSI_Next_Free
                             <1>       ;EAX = First free cluster
                             <1>       ;(from FAT32 FSInfo sector)
0000BDB7 89D0                <1>       mov    eax, edx ; FSI_Next_Free (First Free Cluster)
0000BDB9 83F8FF              <1>       cmp    eax, 0FFFFFFFFh ; invalid (unknown) !
0000BDBC 7205                <1>       jb     short loc_gffc_get_first_fat_free_cluster1
                             <1>
                             <1>       ; Start from the 1st cluster of the FAT(32) file system
                             <1> loc_gffc_get_first_fat_free_cluster0:
0000BDBE B802000000          <1>       mov    eax, 2
                             <1>       ;xor   edx, edx
                             <1>
                             <1> loc_gffc_get_first_fat_free_cluster1:
0000BDC3 53                  <1>       push   ebx ; 02/03/2016
                             <1>
                             <1> loc_gffc_get_first_fat_free_cluster2:
0000BDC4 A3[AC630100]        <1>       mov    [gffc_first_free_cluster], eax
0000BDC9 A3[A8630100]        <1>       mov    [gffc_next_free_cluster], eax
                             <1>
                             <1>       ; EBX = FAT32 FSINFO sector buffer address
                             <1>       ; (EBX = 0, if the drive has not got FAT32 fs or
                             <1>       ; FAT32 FSINFO sector buffer is invalid.)
                             <1>
                             <1> loc_gffc_get_first_fat_free_cluster3:
0000BDCE E875FDFFFF          <1>       call   get_next_cluster
0000BDD3 7307                <1>       jnc    short loc_gffc_get_first_fat_free_cluster4
0000BDD5 09C0                <1>       or     eax, eax
0000BDD7 740B                <1>       jz     short loc_gffc_first_free_fat_cluster_next
```

```
0000BDD9 5B                  <1>       pop    ebx ; 02/03/2016
0000BDDA F5                  <1>       cmc    ; stc
0000BDDB C3                  <1>       retn
                             <1>
                             <1> loc_gffc_get_first_fat_free_cluster4:
0000BDDC 21C0                <1>       and    eax, eax ; next cluster value
0000BDDE 7504                <1>       jnz    short loc_gffc_first_free_fat_cluster_next
0000BDE0 89C8                <1>       mov    eax, ecx ; current (previous cluster) value
0000BDE2 EB22                <1>       jmp    short loc_gffc_check_for_set
                             <1>
                             <1> loc_gffc_first_free_fat_cluster_next:
0000BDE4 A1[A8630100]        <1>       mov    eax, [gffc_next_free_cluster]
0000BDE9 3B05[B0630100]      <1>       cmp    eax, [gffc_last_free_cluster]
0000BDEF 7308                <1>       jnb    short retn_stc_from_get_first_free_cluster
                             <1> pass_gffc_last_cluster_eax_check:
0000BDF1 40                  <1>       inc    eax ; add eax, 1
0000BDF2 A3[A8630100]        <1>       mov    [gffc_next_free_cluster], eax
0000BDF7 EBD5                <1>       jmp    short loc_gffc_get_first_fat_free_cluster3
                             <1>
                             <1> retn_stc_from_get_first_free_cluster:
0000BDF9 A1[AC630100]        <1>       mov    eax, [gffc_first_free_cluster]
0000BDFE 83F802              <1>       cmp    eax, 2
0000BE01 7709                <1>       ja     short loc_gffc_check_previous_clusters
0000BE03 29C0                <1>       sub    eax, eax
0000BE05 48                  <1>       dec    eax ; FFFFFFFFh
                             <1>
                             <1> loc_gffc_check_for_set:
                             <1>       ; 02/03/2016
0000BE06 5B                  <1>       pop    ebx
                             <1>
                             <1>       ; EBX = FAT32 FSINFO sector buffer address
                             <1>       ; (EBX = 0, if the drive has not got FAT32 fs or
                             <1>       ; FAT32 FSINFO sector buffer is invalid.)
                             <1>
0000BE07 09DB                <1>       or     ebx, ebx
0000BE09 750E                <1>       jnz    short loc_gffc_set_ffree_fat32_cluster
                             <1>
                             <1>       ;cmp   byte [esi+LD_FATType], 3
                             <1>       ;jnb   short loc_gffc_set_ffree_fat32_cluster
                             <1>
                             <1>       ;xor   ebx, ebx ; 0
                             <1>
                             <1> loc_gffc_retn:
0000BE0B C3                  <1>       retn
                             <1>
                             <1> loc_gffc_check_previous_clusters:
0000BE0C 48                  <1>       dec    eax ; sub eax, 1
0000BE0D A3[B0630100]        <1>       mov    [gffc_last_free_cluster], eax
0000BE12 B802000000          <1>       mov    eax, 2
                             <1>       ;xor   edx, edx
0000BE17 EBAB                <1>       jmp    short loc_gffc_get_first_fat_free_cluster2
                             <1>
                             <1> loc_gffc_set_ffree_fat32_cluster:
                             <1>       ;call  set_first_free_cluster
                             <1>       ;retn
                             <1>       ;jmp   short set_first_free_cluster
                             <1>
                             <1> set_first_free_cluster:
                             <1>       ; 15/10/2016
                             <1>       ; 23/03/2016
                             <1>       ; 02/03/2016
                             <1>       ; 29/02/2016
                             <1>       ; 26/02/2016
                             <1>       ; 21/02/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>       ; 21/08/2011 (DRV_FAT.ASM, 'proc_set_first_free_cluster')
                             <1>       ; 11/07/2010
                             <1>       ; INPUT ->
                             <1>       ;      ESI = Logical DOS Drive Description Table address
                             <1>       ;      EAX = First free cluster
                             <1>       ;      EBX = FSINFO sector buffer address
                             <1>       ;      ;;If EBX > 0, it is FSINFO sector buffer address
                             <1>       ;      ;;EBX = 0, if FSINFO sector is not loaded
                             <1>       ; OUTPUT->
                             <1>       ;      ESI = Logical DOS Drive Description Table address
                             <1>       ;      If EBX > 0, it is FSINFO sector buffer address
                             <1>       ;      EBX = 0, if FSINFO sector could not be loaded
                             <1>       ;      CF = 1 -> Error code in AL (EAX)
                             <1>       ;      CF = 0 -> first free cluster is successfully updated
                             <1>
                             <1>       ;cmp   byte [esi+LD_FATType], 3
                             <1>       ;jb    short loc_sffc_invalid_drive
                             <1>
                             <1>       ; Save First Free Cluster value for 'update_cluster'
0000BE19 89463E              <1>       mov    [esi+LD_BPB+BPB_Reserved+4], eax ; First free Cluster
                             <1>
                             <1>       ;or    ebx, ebx
                             <1>       ;jnz   short loc_sffc_read_fsinfo_sector
                             <1>
0000BE1C 813B52526141        <1>       cmp     dword [ebx], 41615252h
0000BE22 7540                <1>       jne    short loc_sffc_read_fsinfo_sector
0000BE24 81BBE4010000727241- <1>       cmp    dword [ebx+484], 61417272h
0000BE2D 61                  <1>
0000BE2E 7534                <1>       jne    short loc_sffc_read_fsinfo_sector
                             <1>
0000BE30 3B83EC010000        <1>       cmp    eax, [ebx+492]  ; FSI_Next_Free
0000BE36 741F                <1>       je     short loc_sffc_retn
                             <1>
                             <1> loc_sffc_write_fsinfo_sector:
                             <1>       ; EBX = FSINFO sector buffer
                             <1>       ; [CFS_FAT32FSINFOSEC] is set in 'get_fat32_fsinfo_sector_parms'
0000BE38 8983EC010000        <1>       mov    [ebx+492], eax
0000BE3E A1[C0630100]        <1>       mov    eax, [CFS_FAT32FSINFOSEC]
0000BE43 B901000000          <1>       mov    ecx, 1
```

```
0000BE48 53                  <1>       push   ebx
0000BE49 E87B390000          <1>       call   disk_write
0000BE4E 7208                <1>       jc      short loc_sffc_read_fsinfo_sector_err1
0000BE50 5B                  <1>       pop    ebx
                             <1> 
0000BE51 8B83EC010000        <1>       mov    eax, [ebx+492] ; First (Next) Free Cluster
                             <1> 
                             <1> loc_sffc_retn:
0000BE57 C3                  <1>       retn
                             <1> 
                             <1> ;loc_sffc_invalid_drive:
                             <1> ;     mov    eax, 0Fh ; MSDOS Error : Invalid drive
                             <1> ;     push   edx
                             <1> 
                             <1> loc_sffc_read_fsinfo_sector_err1:
0000BE58 BB00000000          <1>       mov    ebx, 0
                             <1>       ; 15/10/2016 (1Dh -> 18)
                             <1>       ; 23/03/2016 (1Dh)
0000BE5D B812000000          <1>       mov    eax, 18 ; Drive not ready or write error
                             <1> 
                             <1> loc_sffc_read_fsinfo_sector_err2:
0000BE62 5A                  <1>       pop    edx
0000BE63 C3                  <1>       retn
                             <1> 
                             <1> loc_sffc_read_fsinfo_sector:
0000BE64 50                  <1>       push   eax
                             <1> 
0000BE65 E88F050000          <1>       call   get_fat32_fsinfo_sector_parms
0000BE6A 72F6                <1>       jc     short loc_sffc_read_fsinfo_sector_err2
                             <1> 
0000BE6C 58                  <1>       pop    eax
                             <1>       ; EDX = First (Next) Free Cluster value from FSINFO sector
                             <1>       ; EAX = First Free Cluster value from 'get_next_cluster'
                             <1>       ; (edx = old value)
0000BE6D 39D0                <1>       cmp    eax, edx ; First free Cluster (eax = new value)
0000BE6F 75C7                <1>       jne    short loc_sffc_write_fsinfo_sector
                             <1> 
0000BE71 C3                  <1>       retn
                             <1> 
                             <1> update_cluster:
                             <1>       ; 23/10/2016
                             <1>       ; 23/03/2016
                             <1>       ; 02/03/2016
                             <1>       ; 01/03/2016
                             <1>       ; 29/02/2016
                             <1>       ; 27/02/2016
                             <1>       ; 26/02/2016
                             <1>       ; 22/02/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>       ; 11/08/2011
                             <1>       ; 09/02/2005
                             <1>       ; INPUT ->
                             <1>       ;      EAX = Cluster Number
                             <1>       ;      ECX = New Cluster Value
                             <1>       ;      ESI = Logical Dos Drive Parameters Table
                             <1>       ;
                             <1>       ;      /// dword [FAT_ClusterCounter] ///
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      cf = 0 -> No Error, EAX is valid
                             <1>       ;      cf = 1 & EAX = 0 -> End Of Cluster Chain
                             <1>       ;      cf = 1 & EAX > 0 -> Error
                             <1>       ;             (ECX -> any value)
                             <1>       ;      EAX = Next Cluster
                             <1>       ;      ECX = New Cluster Value
                             <1>       ;
                             <1>       ;      /// [FAT_ClusterCounter] is updated,
                             <1>       ;      /// decreased when a free cluster is assigned,
                             <1>       ;      /// increased if an assigned cluster is freed.
                             <1>       ;
                             <1>       ;
                             <1>       ; (Modified registers: EAX, EBX, -ECX-, EDX)
                             <1> 
0000BE72 A3[12610100]        <1>       mov    [FAT_CurrentCluster], eax
0000BE77 890D[B4630100]      <1>       mov    [ClusterValue], ecx
                             <1> 
                             <1> loc_update_cluster_check_fat_buffer:
0000BE7D 8A1E                <1>       mov    bl, [esi+LD_Name]
0000BE7F 381D[17610100]      <1>       cmp    [FAT_BuffDrvName], bl
0000BE85 741A                <1>       je     short loc_update_cluster_check_fat_type
0000BE87 803D[16610100]02    <1>       cmp    byte [FAT_BuffValidData], 2
0000BE8E 0F84C2000000        <1>         je      loc_uc_save_fat_buffer
                             <1> 
                             <1> loc_uc_reset_fat_buffer_validation:
0000BE94 C605[16610100]00    <1>       mov    byte [FAT_BuffValidData], 0
                             <1> 
                             <1> loc_uc_check_fat_type_reset_drvname:
0000BE9B 881D[17610100]      <1>       mov    [FAT_BuffDrvName], bl
                             <1> 
                             <1> loc_update_cluster_check_fat_type:
0000BEA1 29D2                <1>       sub    edx, edx ; 26/02/2016
0000BEA3 8A5E03              <1>       mov    bl, [esi+LD_FATType]
0000BEA6 83F802              <1>       cmp    eax, 2
0000BEA9 0F82BE000000        <1>         jb      update_cluster_inv_data
0000BEAF 80FB02              <1>       cmp    bl, 2
0000BEB2 0F877A010000        <1>         ja      update_fat32_cluster
                             <1>       ;cmp   bl, 1
                             <1>       ;jb    short update_cluster_inv_data
0000BEB8 8B4E78              <1>       mov    ecx, [esi+LD_Clusters]
0000BEBB 41                  <1>       inc    ecx
0000BEBC 890D[22610100]      <1>       mov    [LastCluster], ecx
0000BEC2 39C8                <1>       cmp    eax, ecx ; dword [LastCluster]
0000BEC4 0F87A6000000        <1>         ja      return_uc_fat_stc
                             <1>       ; TRDOS v1 has a FATal bug here !
                             <1>              ; or bl, bl ; cmp bl, 0
                             <1>              ; jz short update_fat12_cluster
                             <1>       ; !! It would destroy FAT12 floppy disk fs here !!
                             <1>       ; ('A:' disks of TRDOS v1 operating system project
                             <1>       ; had 'singlix fs', so, I could not differ this mistake
                             <1>       ; on a drive 'A:')
0000BECA 80FB01              <1>       cmp    bl, 1 ; correct comparison is this !
0000BECD 0F86A2000000        <1>         jna     update_fat12_cluster
                             <1>
                             <1> update_fat16_cluster:
                             <1> pass_uc_fat16_errc:
                             <1>       ;sub   edx, edx
0000BED3 BB00030000          <1>       mov    ebx, 300h ;768
0000BED8 F7F3                <1>       div    ebx
                             <1>       ; EAX = Count of 3 FAT sectors
                             <1>       ; DX = Cluster offset in FAT buffer
0000BEDA 6689D3              <1>       mov    bx, dx
0000BEDD 66D1E3              <1>       shl    bx, 1 ; Multiply by 2
0000BEE0 66BA0300            <1>       mov    dx, 3
0000BEE4 F7E2                <1>       mul    edx
                             <1>       ; EAX = FAT Sector
                             <1>       ; EDX = 0
                             <1>       ; EBX = Byte offset in FAT buffer
0000BEE6 8A0D[16610100]      <1>       mov    cl, [FAT_BuffValidData]
0000BEEC 80F902              <1>       cmp    cl, 2
0000BEEF 750A                <1>       jne    short loc_uc_check_fat16_buff_sector_load
                             <1>
                             <1> loc_uc_check_fat16_buff_sector_save:
0000BEF1 3B05[1A610100]      <1>       cmp    eax, [FAT_BuffSector]
0000BEF7 755D                <1>       jne    short loc_uc_save_fat_buffer
0000BEF9 EB15                <1>       jmp    short loc_update_fat16_cell
                             <1>
                             <1> loc_uc_check_fat16_buff_sector_load:
0000BEFB 80F901              <1>       cmp    cl, 1 ; byte [FAT_BuffValidData]
0000BEFE 0F85FB010000        <1>         jne     loc_uc_load_fat_sectors
0000BF04 3B05[1A610100]      <1>       cmp    eax, [FAT_BuffSector]
0000BF0A 0F85EF010000        <1>         jne     loc_uc_load_fat_sectors
                             <1>
                             <1> loc_update_fat16_cell:
                             <1> loc_update_fat16_buffer:
0000BF10 81C3001C0900        <1>       add    ebx, FAT_Buffer ; 26/02/2016
                             <1>       ;movzx eax, word [ebx]
0000BF16 668B03              <1>       mov    ax, [ebx]
                             <1>       ; 01/03/2016
0000BF19 89C2                <1>       mov    edx, eax ; old value of the cluster
0000BF1B A3[12610100]        <1>       mov    [FAT_CurrentCluster], eax
0000BF20 8B0D[B4630100]      <1>       mov    ecx, [ClusterValue] ; 32 bits
0000BF26 66890B              <1>       mov    [ebx], cx ; 16 bits !
                             <1>
0000BF29 C605[16610100]02    <1>       mov    byte [FAT_BuffValidData], 2
                             <1>
0000BF30 6683F802            <1>       cmp    ax, 2
0000BF34 723A                <1>       jb     short return_uc_fat_stc
0000BF36 3B05[22610100]      <1>       cmp    eax, [LastCluster]
0000BF3C 7732                <1>       ja     short return_uc_fat_stc
                             <1>
                             <1> loc_fat_buffer_updated:
                             <1>       ; 01/03/2016
0000BF3E F8                  <1>       clc
                             <1> loc_fat_buffer_stc_1:
0000BF3F 9C                  <1>       pushf
0000BF40 21C9                <1>       and    ecx, ecx
0000BF42 7506                <1>       jnz    short loc_fat_buffer_updated_1
                             <1>
                             <1>       ; 01/03/2016
                             <1>       ; new value of the cluster = 0 (free)
                             <1>       ; increase free(d) cluster count
0000BF44 FF05[1E610100]      <1>       inc    dword [FAT_ClusterCounter]
                             <1>
                             <1> loc_fat_buffer_updated_1: ; new value of the cluster > 0
0000BF4A 09D2                <1>       or     edx, edx ; 02/03/2016
0000BF4C 7506                <1>       jnz    short loc_fat_buffer_updated_2
                             <1>       ; old value of the cluster = 0 (it was free cluster)
                             <1>       ; decrease free(d) cluster count
0000BF4E FF0D[1E610100]      <1>       dec    dword [FAT_ClusterCounter] ; it may be negative number
                             <1>
                             <1> loc_fat_buffer_updated_2:
0000BF54 9D                  <1>       popf
0000BF55 C3                  <1>       retn
                             <1>
                             <1> loc_uc_save_fat_buffer:
                             <1>       ; byte [FAT_BuffValidData] = 2
0000BF56 E8D4010000          <1>       call   save_fat_buffer
0000BF5B 0F8297010000        <1>         jc      loc_fat_sectors_rw_error2
                             <1>       ;mov   byte [FAT_BuffValidData], 1
0000BF61 A1[12610100]        <1>       mov    eax, [FAT_CurrentCluster]
                             <1>       ;mov   ecx, [ClusterValue]
                             <1>       ;jmp   short loc_update_cluster_check_fat_buffer
0000BF66 8A1E                <1>       mov    bl, [esi+LD_Name] ; 01/03/2016
0000BF68 E927FFFFFF          <1>         jmp     loc_uc_reset_fat_buffer_validation
                             <1>
                             <1> update_cluster_inv_data:
                             <1>       ;mov   eax, 0Dh
0000BF6D B00D                <1>       mov    al, 0Dh  ; Invalid Data
0000BF6F C3                  <1>       retn
                             <1>
                             <1> return_uc_fat_stc:
                             <1>       ; 01/03/2016
0000BF70 31C0                <1>       xor    eax, eax
0000BF72 F9                  <1>       stc
0000BF73 EBCA                <1>       jmp    short loc_fat_buffer_stc_1
                             <1>
                             <1> update_fat12_cluster:
```

```
                             <1> pass_uc_fat12_errc:
                             <1>       ;sub   edx, edx
0000BF75 BB00040000          <1>       mov    ebx, 400h ;1024
0000BF7A F7F3                <1>       div    ebx
                             <1>       ; EAX = Count of 3 FAT sectors
                             <1>       ; DX = Cluster offset in FAT buffer
0000BF7C 66B90300            <1>       mov    cx, 3
0000BF80 6689C3              <1>       mov    bx, ax
0000BF83 6689C8              <1>       mov    ax, cx ; 3
0000BF86 66F7E2              <1>       mul    dx     ; Multiply by 3
0000BF89 66D1E8              <1>       shr    ax, 1  ; Divide by 2
0000BF8C 6693                <1>       xchg   bx, ax
                             <1>       ; EAX = Count of 3 FAT sectors
                             <1>       ; EBX = Byte Offset in FAT buffer
0000BF8E 66F7E1              <1>       mul    cx  ; 3 * AX
                             <1>       ; EAX = FAT Beginning Sector
                             <1>       ; EDX = 0
0000BF91 8A0D[16610100]      <1>       mov    cl, [FAT_BuffValidData]
                             <1>       ; TRDOS v1 has a FATal bug here !
                             <1>       ; (it does not have 'cmp cl, 2' instruction here !
                             <1>       ;  while 'jne' is existing !)
0000BF97 80F902              <1>       cmp    cl, 2 ; 2 = dirty buffer (must be written to disk)
0000BF9A 750A                <1>       jne    short loc_uc_check_fat12_buff_sector_load
                             <1> 
                             <1> loc_uc_check_fat12_buff_sector_save:
0000BF9C 3B05[1A610100]      <1>       cmp    eax, [FAT_BuffSector]
0000BFA2 75B2                <1>         jne     short loc_uc_save_fat_buffer
0000BFA4 EB15                <1>       jmp    short loc_update_fat12_cell
                             <1> 
                             <1> loc_uc_check_fat12_buff_sector_load:
0000BFA6 80F901              <1>       cmp    cl, 1 ; byte ptr [FAT_BuffValidData]
0000BFA9 0F8550010000        <1>         jne     loc_uc_load_fat_sectors
0000BFAF 3B05[1A610100]      <1>       cmp    eax, [FAT_BuffSector]
0000BFB5 0F8544010000        <1>         jne     loc_uc_load_fat_sectors
                             <1> 
                             <1> loc_update_fat12_cell:
0000BFBB 81C3001C0900        <1>       add    ebx, FAT_Buffer ; 26/02/2016
0000BFC1 668B0D[12610100]    <1>       mov    cx, [FAT_CurrentCluster]
0000BFC8 66D1E9              <1>       shr    cx, 1
0000BFCB 668B03              <1>       mov    ax, [ebx]
0000BFCE 6689C2              <1>       mov    dx, ax
0000BFD1 7344                <1>       jnc    short uc_fat12_nc_even
                             <1> 
0000BFD3 6683E00F            <1>       and    ax, 0Fh
0000BFD7 8B0D[B4630100]      <1>       mov    ecx, [ClusterValue] ; 32 bits
0000BFDD 66C1E104            <1>       shl    cx, 4
0000BFE1 6609C1              <1>       or     cx, ax
0000BFE4 6689D0              <1>       mov    ax, dx
0000BFE7 66890B              <1>       mov    [ebx], cx  ; 16 bits !
0000BFEA 66C1E804            <1>       shr    ax, 4 ; al(bit4..7)+ah(bit0..7)
                             <1> 
                             <1> update_fat12_buffer:
0000BFEE A3[12610100]        <1>       mov    [FAT_CurrentCluster], eax
0000BFF3 89C2                <1>       mov    edx, eax ; 01/03/2016
0000BFF5 C605[16610100]02    <1>       mov    byte [FAT_BuffValidData], 2
0000BFFC 6683F802            <1>       cmp    ax, 2
0000C000 0F826AFFFFFF        <1>         jb      return_uc_fat_stc
0000C006 3B05[22610100]      <1>       cmp    eax, [LastCluster]
0000C00C 0F875EFFFFFF        <1>         ja      return_uc_fat_stc
0000C012 E927FFFFFF          <1>         jmp     loc_fat_buffer_updated
                             <1> 
                             <1> uc_fat12_nc_even:
0000C017 662500F0            <1>       and    ax, 0F000h
0000C01B 8B0D[B4630100]      <1>       mov    ecx, [ClusterValue] ; 32 bits
0000C021 80E50F              <1>       and    ch, 0Fh
0000C024 6609C1              <1>       or     cx, ax
0000C027 6689D0              <1>       mov    ax, dx
0000C02A 66890B              <1>       mov    [ebx], cx ; 16 bits !
0000C02D 80E40F              <1>       and    ah, 0Fh ; al(bit0..7)+ah(bit0..3)
0000C030 EBBC                <1>       jmp    short update_fat12_buffer
                             <1> 
                             <1> update_fat32_cluster:
0000C032 8B4E78              <1>       mov    ecx, [esi+LD_Clusters]
0000C035 41                  <1>       inc    ecx
0000C036 890D[22610100]      <1>       mov    [LastCluster], ecx
                             <1> 
0000C03C 39C8                <1>       cmp    eax, ecx
0000C03E 0F872CFFFFFF        <1>         ja      return_uc_fat_stc
                             <1> 
                             <1> pass_uc_fat32_errc:
                             <1>       ;sub   edx, edx
0000C044 BB80010000          <1>       mov    ebx, 180h ;384
0000C049 F7F3                <1>       div    ebx
                             <1>       ; EAX = Count of 3 FAT sectors
                             <1>       ; DX = Cluster offset in FAT buffer
0000C04B 89D3                <1>       mov    ebx, edx
0000C04D C1E302              <1>       shl    ebx, 2 ; Multiply by 4
0000C050 BA03000000          <1>       mov    edx, 3
0000C055 F7E2                <1>       mul    edx
                             <1>       ; EBX = Cluster Offset in FAT buffer
                             <1>       ; EAX = FAT Sector
                             <1>       ; EDX = 0
0000C057 8A0D[16610100]      <1>       mov    cl, [FAT_BuffValidData]
0000C05D 80F902              <1>       cmp    cl, 2
0000C060 750E                <1>       jne    short loc_uc_check_fat32_buff_sector_load
                             <1> 
                             <1> loc_uc_check_fat32_buff_sector_save:
0000C062 3B05[1A610100]      <1>       cmp    eax, [FAT_BuffSector]
0000C068 0F85E8FEFFFF        <1>         jne     loc_uc_save_fat_buffer
0000C06E EB11                <1>       jmp    short loc_update_fat32_cell
                             <1> 
                             <1> loc_uc_check_fat32_buff_sector_load:
0000C070 80F901              <1>       cmp    cl, 1 ; byte [FAT_BuffValidData]
```

```
0000C073 0F8586000000        <1>         jne     loc_uc_load_fat_sectors
0000C079 3B05[1A610100]      <1>       cmp    eax, [FAT_BuffSector]
0000C07F 757E                <1>         jne     loc_uc_load_fat_sectors
                             <1> 
                             <1> loc_update_fat32_cell:
                             <1> loc_update_fat32_buffer:
0000C081 81C3001C0900        <1>       add    ebx, FAT_Buffer ; 26/02/2016
0000C087 8B03                <1>       mov    eax, [ebx]
0000C089 25FFFFFF0F          <1>       and    eax, 0FFFFFFFh ; 28 bit cluster value
                             <1> 
0000C08E 8B15[12610100]      <1>       mov    edx, [FAT_CurrentCluster] ; 01/03/2016
                             <1> 
0000C094 A3[12610100]        <1>       mov    [FAT_CurrentCluster], eax
0000C099 8B0D[B4630100]      <1>       mov    ecx, [ClusterValue]
0000C09F 890B                <1>       mov    [ebx], ecx ; 29/02/2016
                             <1> 
0000C0A1 C605[16610100]02    <1>       mov    byte [FAT_BuffValidData], 2
                             <1> 
                             <1>       ; 01/03/2016
0000C0A8 21C0                <1>       and    eax, eax ; was it free cluster ?
0000C0AA 7514                <1>       jnz    short loc_upd_fat32_c0
                             <1> 
                             <1>       ;or    ecx, ecx ; it will be left free ?!
                             <1>       ;jz    short loc_upd_fat32_c3
                             <1> 
0000C0AC 3B563E              <1>       cmp    edx, [esi+LD_BPB+BPB_Reserved+4] ; First free cluster
0000C0AF 7520                <1>       jne    short loc_upd_fat32_c3
                             <1> 
0000C0B1 3B15[22610100]      <1>       cmp    edx, [LastCluster]
0000C0B7 7207                <1>       jb     short loc_upd_fat32_c0
                             <1> 
0000C0B9 BA02000000          <1>       mov    edx, 2 ; rewind !
0000C0BE EB0E                <1>       jmp    short loc_upd_fat32_c2
                             <1> 
                             <1> loc_upd_fat32_c0:
0000C0C0 FF463E              <1>       inc    dword [esi+LD_BPB+BPB_Reserved+4] ; set it to next cluster
0000C0C3 EB0C                <1>       jmp    short loc_upd_fat32_c3
                             <1> 
                             <1> loc_upd_fat32_c1:
0000C0C5 09C9                <1>       or     ecx, ecx ; will it be free cluster ?
0000C0C7 7508                <1>       jnz    short loc_upd_fat32_c3
                             <1> 
0000C0C9 3B563E              <1>       cmp    edx, [esi+LD_BPB+BPB_Reserved+4] ; First free cluster
0000C0CC 7303                <1>       jnb    short loc_upd_fat32_c3
                             <1> 
                             <1> loc_upd_fat32_c2:
0000C0CE 89563E              <1>       mov    [esi+LD_BPB+BPB_Reserved+4], edx
                             <1> 
                             <1> loc_upd_fat32_c3:
0000C0D1 89C2                <1>       mov    edx, eax
                             <1> 
                             <1> loc_upd_fat32_c4:
0000C0D3 83F802              <1>       cmp    eax, 2
0000C0D6 0F8294FEFFFF        <1>         jb      return_uc_fat_stc
                             <1> 
                             <1> pass_uc_fat32_c_zero_check_2:
0000C0DC 3B05[22610100]      <1>       cmp    eax, [LastCluster]
0000C0E2 0F8788FEFFFF        <1>         ja      return_uc_fat_stc
                             <1> 
0000C0E8 E951FEFFFF          <1>       jmp     loc_fat_buffer_updated
                             <1> 
                             <1> loc_fat_sectors_rw_error1:
                             <1>       ;mov   byte [FAT_BuffValidData], 0
                             <1>       ; 23/10/2016 (15h -> 17)
                             <1>       ; 23/03/2016
0000C0ED B811000000          <1>       mov    eax, 17 ; Drive not ready or read error
0000C0F2 8825[16610100]      <1>       mov    [FAT_BuffValidData], ah ; 0
                             <1> 
                             <1> loc_fat_sectors_rw_error2:
                             <1>       ;mov   eax, error code
                             <1>       ;mov   edx, 0
0000C0F8 8B0D[B4630100]      <1>       mov    ecx, [ClusterValue]
0000C0FE C3                  <1>       retn
                             <1> 
                             <1> loc_uc_load_fat_sectors:
0000C0FF A3[1A610100]        <1>       mov    [FAT_BuffSector], eax
                             <1> 
                             <1> load_uc_fat_sectors_zero:
0000C104 034660              <1>       add    eax, [esi+LD_FATBegin]
0000C107 BB001C0900          <1>       mov    ebx, FAT_Buffer
0000C10C B903000000          <1>       mov    ecx, 3
0000C111 E8C2360000          <1>       call   disk_read
0000C116 72D5                <1>       jc     short loc_fat_sectors_rw_error1
                             <1> 
0000C118 C605[16610100]01    <1>         mov     byte [FAT_BuffValidData], 1
0000C11F A1[12610100]        <1>       mov    eax, [FAT_CurrentCluster]
0000C124 8B0D[B4630100]      <1>       mov    ecx, [ClusterValue]
0000C12A E972FDFFFF          <1>         jmp     loc_update_cluster_check_fat_type
                             <1> 
                             <1> save_fat_buffer:
                             <1>       ; 15/10/2016
                             <1>       ; 01/03/2016
                             <1>       ; 22/02/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>       ; 11/08/2011
                             <1>       ; 09/02/2005
                             <1>       ; INPUT ->
                             <1>       ;      None
                             <1>       ; OUTPUT ->
                             <1>       ;      cf = 0 -> OK.
                             <1>       ;      cf = 1 -> error code in AL (EAX)
                             <1>       ;
                             <1>       ;      EBX = FAT_Buffer address
                             <1>       ;
```

```
                                    <1>       ; (EAX, EDX, ECX will be modified)
                                    <1>
                                    <1>       ;cmp   byte [FAT_BuffValidData], 2
                                    <1>       ;je    short loc_save_fat_buff
                                    <1>
                                    <1> ;loc_save_fat_buffer_retn:
                                    <1> ;     xor    eax, eax
                                    <1> ;     retn
                                    <1>
                                    <1> loc_save_fat_buff:
0000C12F 31D2                       <1>       xor    edx, edx
0000C131 8A35[17610100]             <1>       mov    dh, [FAT_BuffDrvName]
0000C137 80FE41                     <1>       cmp    dh, 'A'
0000C13A 722E                       <1>       jb     short loc_save_fat_buffer_inv_data_retn
0000C13C 80EE41                     <1>       sub    dh, 'A'
0000C13F 56                         <1>       push   esi ; *
0000C140 BE00010900                 <1>         mov     esi, Logical_DOSDisks
0000C145 01D6                       <1>       add    esi, edx
                                    <1>
0000C147 8A5603                     <1>       mov    dl, [esi+LD_FATType]
0000C14A 20D2                       <1>       and    dl, dl
0000C14C 741B                       <1>       jz     short loc_save_fat_buffer_inv_data_pop_retn
                                    <1>
0000C14E A1[1A610100]               <1>       mov    eax, [FAT_BuffSector]
0000C153 80FA02                     <1>       cmp    dl, 2
0000C156 770A                       <1>       ja     short loc_save_fat32_buff
                                    <1>
                                    <1> loc_save_fat_12_16_buff:
                                    <1>       ; 01/03/2016
                                    <1>       ; TRDOS v1 has a FATal bug here!
                                    <1>       ; Correct code: mov dx, word ptr [FAT_BuffSector]+2
                                    <1>       ; (DX:AX in TRDOS v1 -> EAX in TRDOS v2)
                                    <1>       ;
0000C158 0FB74E1C                   <1>       movzx  ecx, word [esi+LD_BPB+FATSecs]
0000C15C 29C1                       <1>       sub    ecx, eax
                                    <1>       ; TRDOS v1 has a bug here... ('pop esi' was forgotten!)
                                    <1>       ;jna   short loc_save_fat_buffer_inv_data_retn ; wrong addr!
0000C15E 7609                       <1>       jna    short loc_save_fat_buffer_inv_data_pop_retn ; correct addr.
0000C160 EB15                       <1>       jmp    short loc_save_fat_buffer_check_rs3
                                    <1>
                                    <1> loc_save_fat32_buff:
0000C162 8B4E2A                     <1>       mov    ecx, [esi+LD_BPB+FAT32_FAT_Size]
0000C165 29C1                       <1>       sub    ecx, eax
0000C167 770E                       <1>       ja     short loc_save_fat_buffer_check_rs3
                                    <1>
                                    <1> loc_save_fat_buffer_inv_data_pop_retn:
0000C169 5E                         <1>       pop    esi ; *
                                    <1> loc_save_fat_buffer_inv_data_retn:
0000C16A B80D000000                 <1>       mov    eax, 0Dh ; Invalid DATA
0000C16F C3                         <1>       retn
                                    <1>
                                    <1> loc_save_fat_buff_remain_sectors_3:
0000C170 B903000000                 <1>       mov    ecx, 3
0000C175 EB05                       <1>       jmp    short loc_save_fat_buff_continue
                                    <1>
                                    <1> loc_save_fat_buffer_check_rs3:
0000C177 83F903                     <1>       cmp    ecx, 3
0000C17A 77F4                       <1>       ja     short loc_save_fat_buff_remain_sectors_3
                                    <1>
                                    <1> loc_save_fat_buff_continue:
0000C17C BB001C0900                 <1>       mov    ebx, FAT_Buffer
0000C181 034660                     <1>       add    eax, [esi+LD_FATBegin]
0000C184 51                         <1>       push   ecx
0000C185 E83F360000                 <1>       call   disk_write
0000C18A 59                         <1>       pop    ecx
0000C18B 722B                       <1>       jc     short loc_save_FAT_buff_write_err
                                    <1>
0000C18D 807E0302                   <1>       cmp    byte [esi+LD_FATType], 2
0000C191 7605                       <1>       jna    short loc_calc_2nd_fat12_16_addr
                                    <1>
                                    <1> loc_calc_2nd_fat32_addr:
0000C193 8B462A                     <1>       mov    eax, [esi+LD_BPB+FAT32_FAT_Size]
0000C196 EB04                       <1>       jmp    short loc_calc_2nd_fat_addr
                                    <1>
                                    <1> loc_calc_2nd_fat12_16_addr:
0000C198 0FB7461C                   <1>       movzx  eax, word [esi+LD_BPB+FATSecs]
                                    <1>
                                    <1> loc_calc_2nd_fat_addr:
0000C19C 034660                     <1>       add    eax, [esi+LD_FATBegin]
0000C19F 0305[1A610100]             <1>       add    eax, [FAT_BuffSector]
0000C1A5 BB001C0900                 <1>       mov    ebx, FAT_Buffer
                                    <1>       ; ecx = 1 to 3
0000C1AA E81A360000                 <1>       call   disk_write
0000C1AF 7207                       <1>       jc     short loc_save_FAT_buff_write_err
                                    <1>       ; Valid  buffer (1 = valid but do not save)
0000C1B1 C605[16610100]01           <1>       mov    byte [FAT_BuffValidData], 1
                                    <1>
                                    <1> loc_save_FAT_buff_write_err:
0000C1B8 5E                         <1>       pop    esi ; *
0000C1B9 BB001C0900                 <1>       mov    ebx, FAT_Buffer
                                    <1>       ; 15/10/2016 (1Dh -> 18)
                                    <1>       ; 23/03/2016 (1Dh)
0000C1BE B812000000                 <1>       mov    eax, 18 ; Drive not ready or write error
0000C1C3 C3                         <1>       retn
                                    <1>
                                    <1> calculate_fat_freespace:
                                    <1>       ; 23/03/2016
                                    <1>       ; 02/03/2016
                                    <1>       ; 01/03/2016
                                    <1>       ; 29/02/2016
                                    <1>       ; 22/02/2016 (TRDOS 386 =  TRDOS v2.0)
                                    <1>       ; 30/04/2011
                                    <1>       ; 03/04/2010
```

```
                             <1>       ; 2005
                             <1>       ; INPUT ->
                             <1>       ;      EAX = Cluster count to be added or subtracted
                             <1>       ;      If BH = FFh, ESI = TR-DOS Logical Drive Description Table
                             <1>       ;      If BH < FFh, BH = TR-DOS Logical Drive Number
                             <1>       ;      BL:
                             <1>       ;      0 = Calculate, 1 = Add, 2 = Subtract, 3 = Get (Not Set/Calc)
                             <1>       ; OUTPUT ->
                             <1>       ;      EAX = Free Space in sectors
                             <1>       ;      ESI = Logical Dos Drive Description Table address
                             <1>       ;      BH = Logical Dos Drive Number (same with input value of BH)
                             <1>       ;      BL = Type of operation (same with input value of BL)
                             <1>       ;      ECX = 0 -> valid
                             <1>       ;      ECX > 0 -> error or invalid
                             <1>       ;      If EAX = FFFFFFFFh, it is 're-calculation needed'
                             <1>       ;                              sign due to r/w error
                             <1>
0000C1C4 66891D[BA630100]    <1>       mov    [CFS_OPType], bx
0000C1CB A3[BC630100]        <1>       mov    [CFS_CC], eax
                             <1>
0000C1D0 80FFFF              <1>       cmp    bh, 0FFh
0000C1D3 740B                <1>       je     short pass_calculate_freespace_get_drive_dt_offset
                             <1>
                             <1> loc_calculate_freespace_get_drive_dt_offset:
0000C1D5 31C0                <1>       xor    eax, eax
0000C1D7 88FC                <1>         mov     ah, bh
0000C1D9 BE00010900          <1>       mov    esi, Logical_DOSDisks
0000C1DE 01C6                <1>         add     esi, eax
                             <1>
                             <1> pass_calculate_freespace_get_drive_dt_offset:
0000C1E0 08DB                <1>       or     bl, bl
0000C1E2 7435                <1>       jz     short loc_reset_fcc
                             <1>
                             <1> loc_get_free_sectors:
0000C1E4 8B4674              <1>       mov    eax, [esi+LD_FreeSectors]
                             <1>
                             <1>       ;xor   ecx, ecx
                             <1>       ;dec   ecx ; 0FFFFFFFFh
                             <1>       ;cmp   eax, ecx ; 29/02/2016
                             <1>       ;je    short loc_get_free_sectors_retn ; recalculation is needed!
                             <1>
                             <1>       ; 23/03/2016
0000C1E7 8B4E70              <1>       mov    ecx, [esi+LD_TotalSectors]
0000C1EA 39C1                <1>       cmp    ecx, eax ; Total sectors must be greater than Free sectors !
0000C1EC 7707                <1>       ja     short loc_get_free_sectors_check_optype
                             <1>
0000C1EE 31C0                <1>       xor    eax, eax
0000C1F0 48                  <1>       dec    eax ; 0FFFFFFFFh  ; recalculation is needed!
0000C1F1 894674              <1>       mov    [esi+LD_FreeSectors], eax ; reset (for recalculation)
                             <1>
                             <1> loc_get_free_sectors_retn:
0000C1F4 C3                  <1>       retn
                             <1>
                             <1> loc_get_free_sectors_check_optype:
0000C1F5 80FB03              <1>       cmp    bl, 3
0000C1F8 7203                <1>       jb     short loc_set_fcc
                             <1>
0000C1FA 29C9                <1>       sub    ecx, ecx ; 0
                             <1>
0000C1FC C3                  <1>       retn
                             <1>
                             <1> loc_set_fcc:
0000C1FD 807E0302            <1>       cmp    byte [esi+LD_FATType], 2
0000C201 0F87DF000000        <1>         ja      loc_update_FAT32_fs_info_fcc
                             <1>
                             <1>       ;mov   eax, [esi+LD_FreeSectors]
0000C207 0FB64E13            <1>       movzx  ecx, byte [esi+LD_BPB+SecPerClust]
0000C20B 29D2                <1>       sub    edx, edx
0000C20D F7F1                <1>       div    ecx
                             <1>       ;or    dx, dx
                             <1>       ;      ; DX -> Remain sectors < SecPerClust
                             <1>       ;      ; DX > 0 -> invalid free sector count
                             <1>       ;jnz   short loc_reset_fcc
                             <1>
                             <1> ;pass_set_fcc_div32:
0000C20F A3[33610100]        <1>       mov    [FreeClusterCount], eax
0000C214 E988000000          <1>         jmp     loc_set_free_sectors_FAT12_FAT16
                             <1>
                             <1> loc_reset_fcc:
0000C219 31C0                <1>       xor    eax, eax
0000C21B A3[33610100]        <1>       mov    [FreeClusterCount], eax ; 0
0000C220 8B5678              <1>       mov    edx, [esi+LD_Clusters]
0000C223 42                  <1>       inc    edx
0000C224 8915[22610100]      <1>       mov    [LastCluster], edx
                             <1>
0000C22A 807E0302            <1>       cmp    byte [esi+LD_FATType], 2
0000C22E 7647                <1>       jna    short loc_count_free_fat_clusters_0
                             <1>
0000C230 48                  <1>       dec    eax ; FFFFFFFFh
0000C231 A3[C4630100]        <1>       mov    [CFS_FAT32FC], eax
                             <1>
                             <1>       ; 29/02/2016
0000C236 89463A              <1>       mov    [esi+LD_BPB+BPB_Reserved], eax ; reset
0000C239 89463E              <1>       mov    [esi+LD_BPB+BPB_Reserved+4], eax ; reset
                             <1>
0000C23C B802000000          <1>       mov    eax, 2
                             <1>
                             <1> loc_count_fc_next_cluster_0:
0000C241 50                  <1>       push   eax
0000C242 E801F9FFFF          <1>       call   get_next_cluster
0000C247 7310                <1>       jnc    short loc_check_fat32_ff_cluster
0000C249 09C0                <1>       or     eax, eax
0000C24B 741E                <1>       jz     short pass_inc_cfs_fcc_0
```

```
                             <1> 
                             <1> loc_put_fcc_unknown_sign:
0000C24D 58                  <1>       pop    eax
                             <1>       ; "Free count is Unknown" sign
                             <1>       ;mov   dword [FreeClusterCount], 0FFFFFFFFh
                             <1> 
                             <1>       ; 29/02/2016
                             <1>       ; Save Free Cluster Count value in FAT32 'BPB_Reserved' area
                             <1>       ;mov   [esi+LD_BPB+BPB_Reserved], 0FFFFFFFFh ; unknown!
0000C24E 8B15[C4630100]      <1>       mov    edx, [CFS_FAT32FC] ; First Free Cluster
                             <1>       ; Save First Free Cluster value in FAT32 'BPB_Reserved+4' area
0000C254 89563E              <1>       mov    [esi+LD_BPB+BPB_Reserved+4], edx
                             <1> 
0000C257 EB7D                <1>         jmp     loc_put_fcc_invalid_sign
                             <1> 
                             <1> loc_check_fat32_ff_cluster:
0000C259 09C0                <1>       or     eax, eax
0000C25B 750E                <1>       jnz    short pass_inc_cfs_fcc_0
0000C25D 58                  <1>       pop    eax
0000C25E A3[C4630100]        <1>       mov    [CFS_FAT32FC], eax
                             <1>       ;mov   dword [FreeClusterCount], 1
0000C263 FF05[33610100]      <1>       inc    dword [FreeClusterCount]
0000C269 EB27                <1>       jmp    short pass_inc_cfs_fcc_1
                             <1> 
                             <1> pass_inc_cfs_fcc_0:
0000C26B 58                  <1>       pop    eax
                             <1> 
                             <1> pass_inc_cfs_fcc_0c:
0000C26C 40                  <1>       inc    eax ; add eax, 1
0000C26D 3B05[22610100]      <1>       cmp    eax, [LastCluster]
0000C273 76CC                <1>       jna    short loc_count_fc_next_cluster_0
0000C275 EB6F                <1>       jmp    short loc_update_FAT32_fs_info_fcc
                             <1> 
                             <1> loc_count_free_fat_clusters_0:
                             <1>       ;mov   eax, 2
0000C277 B002                <1>       mov    al, 2
                             <1> 
                             <1> loc_count_fc_next_cluster:
0000C279 50                  <1>       push   eax
0000C27A E8C9F8FFFF          <1>       call   get_next_cluster
0000C27F 720C                <1>       jc     short loc_count_fcc_stc
                             <1> 
                             <1> loc_count_free_clusters_1:
0000C281 21C0                <1>       and    eax, eax
0000C283 750C                <1>       jnz    short pass_inc_cfs_fcc
                             <1> 
0000C285 FF05[33610100]      <1>       inc    dword [FreeClusterCount]
0000C28B EB04                <1>       jmp    short pass_inc_cfs_fcc
                             <1> 
                             <1> loc_count_fcc_stc:
0000C28D 09C0                <1>       or     eax, eax
0000C28F 75BC                <1>       jnz    short loc_put_fcc_unknown_sign ; 29/02/2016
                             <1> 
                             <1> pass_inc_cfs_fcc:
0000C291 58                  <1>       pop    eax
                             <1> 
                             <1> pass_inc_cfs_fcc_1:
0000C292 40                  <1>       inc    eax ; add eax, 1
0000C293 3B05[22610100]      <1>       cmp    eax, [LastCluster]
0000C299 76DE                <1>       jna    short loc_count_fc_next_cluster
                             <1> 
                             <1> loc_set_free_sectors:
0000C29B 807E0302            <1>       cmp    byte [esi+LD_FATType], 2
0000C29F 7745                <1>       ja     short loc_update_FAT32_fs_info_fcc
                             <1> 
                             <1> loc_set_free_sectors_FAT12_FAT16:
0000C2A1 803D[BA630100]00    <1>       cmp    byte [CFS_OPType], 0
0000C2A8 761C                <1>       jna    short pass_FAT_add_sub_fcc
0000C2AA A1[BC630100]        <1>       mov    eax, [CFS_CC]
0000C2AF 803D[BA630100]01    <1>       cmp    byte [CFS_OPType], 1
0000C2B6 7708                <1>       ja     short pass_FAT_add_fcc
0000C2B8 0105[33610100]      <1>       add    [FreeClusterCount], eax
0000C2BE EB06                <1>       jmp    short pass_FAT_add_sub_fcc
                             <1> 
                             <1> pass_FAT_add_fcc:
0000C2C0 2905[33610100]      <1>       sub    [FreeClusterCount], eax
                             <1> 
                             <1> pass_FAT_add_sub_fcc:
0000C2C6 0FB64613            <1>       movzx  eax, byte [esi+LD_BPB+SecPerClust]
0000C2CA 8B15[33610100]      <1>       mov    edx, [FreeClusterCount]
0000C2D0 F7E2                <1>       mul    edx
                             <1> 
0000C2D2 31C9                <1>       xor    ecx, ecx
0000C2D4 EB05                <1>       jmp    short loc_cfs_retn_params
                             <1> 
                             <1> loc_put_fcc_invalid_sign:
0000C2D6 29C0                <1>              sub    eax, eax ; 0
0000C2D8 48                  <1>       dec    eax ; FFFFFFFFh
                             <1> loc_fat32_ffc_recalc_needed:
0000C2D9 89C1                <1>       mov    ecx, eax
                             <1> 
                             <1> loc_cfs_retn_params:
0000C2DB 894674              <1>       mov    [esi+LD_FreeSectors], eax
0000C2DE 0FB71D[BA630100]    <1>       movzx  ebx, word [CFS_OPType]
0000C2E5 C3                  <1>       retn
                             <1> 
                             <1> loc_update_FAT32_fs_info_fcc:
                             <1> loc_check_fcc_FSINFO_op:
                             <1>       ; 29/02/2016
                             <1>       ; EAX = Free cluster count (before this update) ; value from disk
                             <1>       ; EDX = First Free Cluster (before this update) ; value from disk
0000C2E6 803D[BA630100]01    <1>       cmp    byte [CFS_OPType], 1
0000C2ED 7221                <1>       jb     short loc_cfs_FAT32_get_rcalc_parms ; 0 = recalculated
```

```
0000C2EF 7406                <1>         je     short loc_check_fcc_FSINFO_op1 ; 1 = add
                             <1> loc_check_fcc_FSINFO_op2: ; subtract
0000C2F1 F71D[BC630100]      <1>         neg    dword [CFS_CC] ; prepare to subtract ; 2 = sub (add negative)
                             <1> loc_check_fcc_FSINFO_op1:
                             <1>         ; 01/03/2016
0000C2F7 31D2                <1>         xor    edx, edx ; 0
0000C2F9 4A                  <1>         dec    edx ; 0FFFFFFFFh
0000C2FA 8B463A              <1>         mov    eax, [esi+LD_BPB+BPB_Reserved]
0000C2FD 39D0                <1>         cmp    eax, edx
0000C2FF 73D5                <1>         jnb    short loc_put_fcc_invalid_sign
0000C301 0305[BC630100]      <1>           add     eax, [CFS_CC] ; free cluster count on disk + current count
0000C307 72CD                <1>         jc     short loc_put_fcc_invalid_sign
                             <1> 
0000C309 A3[33610100]        <1>         mov    [FreeClusterCount], eax
0000C30E EB0E                <1>         jmp    short loc_cfs_write_FSINFO_sector
                             <1> 
                             <1> loc_cfs_FAT32_get_rcalc_parms:
0000C310 8B15[C4630100]      <1>         mov    edx, [CFS_FAT32FC]
0000C316 A1[33610100]        <1>         mov    eax, [FreeClusterCount]
0000C31B 89563E              <1>         mov    [esi+LD_BPB+BPB_Reserved+4], edx ; First Free Cluster
                             <1> loc_cfs_write_FSINFO_sector:
0000C31E 89463A              <1>         mov    [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count
                             <1>         ; 01/03/2016
0000C321 E8AA000000          <1>         call   set_fat32_fsinfo_sector_parms
0000C326 72AE                <1>           jc      short loc_put_fcc_invalid_sign
                             <1> 
                             <1> loc_set_FAT32_free_sectors:
                             <1>         ; 29/02/2016
                             <1>         ;mov   eax, [FreeClusterCount]
                             <1>         ;mov   ecx, eax
                             <1>         ;cmp   eax, 0FFFFFFFFh ; Invalid !
                             <1>         ;je    short loc_cfs_retn_params
                             <1>         ;
0000C328 8B0D[33610100]      <1>         mov    ecx, [FreeClusterCount]
0000C32E 0FB64613            <1>         movzx eax, byte [esi+LD_BPB+SecPerClust]
0000C332 F7E1                <1>         mul    ecx
                             <1>         ; 29/02/2016
0000C334 31C9                <1>         xor    ecx, ecx ; 0
0000C336 09D2                <1>         or     edx, edx ; 0 ?
0000C338 759C                <1>           jnz     loc_put_fcc_invalid_sign
0000C33A 394670              <1>         cmp    [esi+LD_TotalSectors], eax ; Volume size in sectors
0000C33D 7697                <1>           jna     short loc_put_fcc_invalid_sign
                             <1>         ;
                             <1> loc_set_FAT32_free_sectors_ok:
0000C33F 31D2                <1>         xor    edx, edx ; 0
0000C341 EB98                <1>           jmp     short loc_cfs_retn_params
                             <1>         ;
                             <1> 
                             <1> get_last_cluster:
                             <1>         ; 22/10/2016
                             <1>         ; 27/02/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>         ; 12/06/2010 (DRV_FAT.ASM, 'proc_get_last_custer')
                             <1>         ; 06/06/2010
                             <1>         ; INPUT ->
                             <1>         ;       EAX = First Cluster Number
                             <1>         ;       ESI = Logical Dos Drive Parameters Table
                             <1>         ; OUTPUT ->
                             <1>         ;       cf = 0 -> No Error, EAX is valid
                             <1>         ;       cf = 1 -> EAX > 0 -> Error
                             <1>         ;       EAX = Last Cluster Number
                             <1>         ;        ECX = Previous Cluster -just before the last cluster-
                             <1>         ;        ; 22/10/2016
                             <1>         ;       [glc_index] = cluster index number of the last cluster
                             <1>         ;
                             <1>         ; (Modified registers: EAX, ECX, EBX, EDX)
                             <1> 
0000C343 89C1                <1>         mov    ecx, eax
                             <1> 
0000C345 C705[CC630100]FFFF- <1>         mov    dword [glc_index], 0FFFFFFFFh ; 22/10/2016
0000C34D FFFF                <1> 
                             <1> 
                             <1> loc_glc_get_next_cluster_1:
0000C34F 890D[C8630100]      <1>         mov    [glc_prevcluster], ecx
                             <1>         ; 22/10/2016
0000C355 FF05[CC630100]      <1>         inc    dword [glc_index]
                             <1> 
                             <1> loc_glc_get_next_cluster_2:
0000C35B E8E8F7FFFF          <1>         call   get_next_cluster
                             <1>         ; ecx = current/previous cluster
                             <1>         ; eax = next/last cluster
0000C360 73ED                <1>         jnc    short loc_glc_get_next_cluster_1
                             <1> 
0000C362 09C0                <1>         or     eax, eax
0000C364 7509                <1>         jnz    short loc_glc_stc_retn
                             <1> 
                             <1>         ; ecx = previous cluster
0000C366 89C8                <1>           mov eax, ecx
                             <1> 
                             <1>         ; previous cluster becomes last cluster (ecx -> eax)
                             <1>         ; previous of previous cluster becomes previous cluster (ecx)
                             <1> 
                             <1> loc_glc_prev_cluster_retn:
0000C368 8B0D[C8630100]      <1>         mov    ecx, [glc_prevcluster]
0000C36E C3                  <1>         retn
                             <1> 
                             <1> loc_glc_stc_retn:
0000C36F F5                  <1>         cmc    ;stc
0000C370 EBF6                <1>           jmp  short loc_glc_prev_cluster_retn
                             <1> 
                             <1> truncate_cluster_chain:
                             <1>         ; 01/03/2016
                             <1>         ; 28/02/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>         ; 22/01/2011 (DRV_FAT.ASM, 'proc_truncate_cluster_chain')
```

```
                              <1>       ; 11/09/2010
                              <1>       ; INPUT ->
                              <1>       ;       ESI = Logical dos drive description table address
                              <1>       ;       EAX = First cluster to be truncated/unlinked
                              <1>       ; OUTPUT ->
                              <1>       ;       ESI = Logical dos drive description table address
                              <1>       ;       ECX = Count of truncated/removed clusters
                              <1>       ;       CF = 0 -> EAX = Free sectors
                              <1>       ;       CF = 1 -> Error code in EAX (AL)
                              <1>
                              <1>       ; NOTE: This procedure does not update lm date&time !
                              <1>
                              <1> loc_truncate_cc:
0000C372 31C9                 <1>       xor    ecx, ecx ; mov ecx, 0
                              <1>       ;mov   byte [FAT_BuffValidData], 0
0000C374 890D[1E610100]       <1>       mov    [FAT_ClusterCounter], ecx ; 0 ; reset
                              <1>
                              <1> loc_tcc_unlink_clusters:
0000C37A E8F3FAFFFF           <1>       call   update_cluster
                              <1>       ; EAX = Next Cluster
                              <1>       ; ECX = Cluster Value
                              <1>       ; Note:
                              <1>       ; Returns count of unlinked clusters in
                              <1>       ; dword ptr FAT_ClusterCounter
0000C37F 73F9                 <1>       jnc short loc_tcc_unlink_clusters
                              <1>
                              <1> pass_tcc_unlink_clusters:
0000C381 A2[D3630100]         <1>       mov    byte [TCC_FATErr], al
0000C386 803D[16610100]02     <1>       cmp    byte [FAT_BuffValidData], 2
0000C38D 750E                 <1>       jne    short loc_tcc_calculate_FAT_freespace
0000C38F E89BFDFFFF           <1>       call   save_fat_buffer
0000C394 7307                 <1>       jnc    short loc_tcc_calculate_FAT_freespace
0000C396 A2[D3630100]         <1>       mov    byte [TCC_FATErr], al ; Error
                              <1>       ;mov   byte [FAT_BuffValidData], 0
                              <1>
                              <1>       ; 01/03/2016
0000C39B EB12                 <1>       jmp    short loc_tcc_recalculate_FAT_freespace
                              <1>
                              <1> loc_tcc_calculate_FAT_freespace:
0000C39D A1[1E610100]         <1>       mov    eax, [FAT_ClusterCounter] ; signed (+-) number
0000C3A2 66BB01FF             <1>       mov    bx, 0FF01h ; BH = FFh -> ESI = Dos drv desc. table
                              <1>                       ; BL = 1 -> add cluster
0000C3A6 E819FEFFFF           <1>       call   calculate_fat_freespace
0000C3AB 21C9                 <1>       and    ecx, ecx ; cx = 0 -> valid free sector count
0000C3AD 7409                 <1>       jz     short pass_truncate_cc_recalc_FAT_freespace
                              <1>
                              <1> loc_tcc_recalculate_FAT_freespace:
0000C3AF 66BB00FF             <1>       mov    bx, 0FF00h ; recalculate !
0000C3B3 E80CFEFFFF           <1>       call   calculate_fat_freespace
                              <1>
                              <1> loc_tcc_calculate_FAT_freespace_err:
                              <1> pass_truncate_cc_recalc_FAT_freespace:
0000C3B8 8B0D[1E610100]       <1>       mov    ecx, [FAT_ClusterCounter]
                              <1>
0000C3BE 803D[D3630100]00     <1>       cmp    byte [TCC_FATErr], 0
0000C3C5 7608                 <1>       jna    short loc_tcc_unlink_clusters_retn
                              <1>
                              <1> loc_tcc_unlink_clusters_error:
0000C3C7 0FB605[D3630100]     <1>       movzx  eax, byte [TCC_FATErr]
0000C3CE F9                   <1>       stc
                              <1> loc_tcc_unlink_clusters_retn:
0000C3CF C3                   <1>       retn
                              <1>
                              <1> set_fat32_fsinfo_sector_parms:
                              <1>       ; 15/10/2016
                              <1>       ; 23/03/2016
                              <1>       ; 29/02/2016 (TRDOS 386 =  TRDOS v2.0)
                              <1>       ; INPUT ->
                              <1>       ;       ESI = Logical dos drive description table address
                              <1>       ;       [esi+LD_BPB+BPB_Reserved] = Free Cluster Count
                              <1>       ;       [esi+LD_BPB+BPB_Reserved+4] = First Free Cluster
                              <1>       ; OUTPUT ->
                              <1>       ;       ESI = Logical dos drive description table address
                              <1>       ;       CF = 0 -> OK..
                              <1>       ;       CF = 1 -> Error code in EAX (AL)
                              <1>       ;
                              <1>       ; (Modified registers: EAX, EBX, ECX, EDX)
                              <1>
0000C3D0 E824000000           <1>       call   get_fat32_fsinfo_sector_parms
0000C3D5 7221                 <1>       jc     short update_fat32_fsinfo_sector_retn
                              <1>
0000C3D7 8B463A               <1>       mov    eax, [esi+LD_BPB+BPB_Reserved] ; Free Cluster Count
0000C3DA 8B563E               <1>       mov    edx, [esi+LD_BPB+BPB_Reserved+4] ; First free Cluster
                              <1>
                              <1>         ;mov ebx, DOSBootSectorBuff
0000C3DD 8983E8010000         <1>       mov    [ebx+488], eax
0000C3E3 8993EC010000         <1>       mov    [ebx+492], edx
                              <1>
0000C3E9 A1[C0630100]         <1>       mov    eax, [CFS_FAT32FSINFOSEC]
0000C3EE B901000000           <1>       mov    ecx, 1
0000C3F3 E8D1330000           <1>       call   disk_write
                              <1>       ;jnc      short update_fat32_fsinfo_sector_retn
                              <1>
                              <1>       ; 15/10/2016 (1Dh -> 18)
                              <1>       ; 23/03/2016 (1Dh)
                              <1>       ;mov   eax, 18 ; Drive not ready or write error
                              <1>
                              <1> update_fat32_fsinfo_sector_retn:
0000C3F8 C3                   <1>       retn
                              <1>
                              <1> get_fat32_fsinfo_sector_parms:
                              <1>       ; 15/10/2016
                              <1>       ; 23/03/2016
```

```
                             <1>       ; 01/03/2016
                             <1>       ; 29/02/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>       ; INPUT ->
                             <1>       ;      ESI = Logical dos drive description table address
                             <1>       ; OUTPUT ->
                             <1>       ;      ESI = Logical dos drive description table address
                             <1>       ;      EBX = FSINFO sector buffer address (DOSBootSectorBuff)
                             <1>       ;      CF = 0 -> OK..
                             <1>       ;         EAX = FsInfo sector address
                             <1>       ;         ECX = Free cluster count
                             <1>       ;         EDX = First free cluster
                             <1>       ;      CF = 1 -> Error code in AL (EAX)
                             <1>       ;         EBX = 0
                             <1>       ;
                             <1>       ;      [CFS_FAT32FSINFOSEC] = FAT32 FSINFO sector address
                             <1>         ;
                             <1>       ; (Modified registers: EAX, EBX, ECX, EDX)
                             <1>
0000C3F9 0FB74636            <1>       movzx  eax, word [esi+LD_BPB+FAT32_FSInfoSec]
0000C3FD 03466C              <1>       add    eax, [esi+LD_StartSector]
0000C400 A3[C0630100]        <1>       mov    [CFS_FAT32FSINFOSEC], eax
                             <1>
0000C405 BB[125F0100]        <1>         mov     ebx, DOSBootSectorBuff
0000C40A B901000000          <1>       mov    ecx, 1
0000C40F E8C4330000          <1>       call   disk_read
0000C414 7232                <1>       jc     short loc_read_FAT32_fsinfo_sec_err
                             <1>
0000C416 BB[125F0100]        <1>       mov    ebx, DOSBootSectorBuff
                             <1>
0000C41B 813B52526141        <1>       cmp    dword [ebx], 41615252h
0000C421 751E                <1>       jne    short loc_read_FAT32_fsinfo_sec_stc
                             <1>
0000C423 81BBE4010000727241- <1>       cmp    dword [ebx+484], 61417272h
0000C42C 61                  <1>
0000C42D 7512                <1>       jne    short loc_read_FAT32_fsinfo_sec_stc
                             <1>
0000C42F A1[C0630100]        <1>       mov    eax, [CFS_FAT32FSINFOSEC]
0000C434 8B8BE8010000        <1>       mov    ecx, [ebx+488] ; free cluster count
0000C43A 8B93EC010000        <1>       mov    edx, [ebx+492] ; first (next) free cluster
                             <1>
0000C440 C3                  <1>       retn
                             <1>
                             <1> loc_read_FAT32_fsinfo_sec_stc:
                             <1>       ; 15/10/2016 (0Bh -> 28)
0000C441 B81C000000          <1>       mov    eax, 28 ; Invalid format!
0000C446 EB05                <1>       jmp    short loc_read_FAT32_fsinfo_sec_stc_retn
                             <1>
                             <1> loc_read_FAT32_fsinfo_sec_err:
                             <1>       ; 15/10/2016 (15h -> 17)
                             <1>       ; 23/03/2016 (15h)
0000C448 B811000000          <1>       mov    eax, 17 ; Drive not ready or read error
                             <1>
                             <1> loc_read_FAT32_fsinfo_sec_stc_retn:
0000C44D 29DB                <1>       sub    ebx, ebx ; 0
0000C44F F9                  <1>       stc
0000C450 C3                  <1>       retn
                             <1>
                             <1> add_new_cluster:
                             <1>       ; 15/10/2016
                             <1>       ; 16/05/2016
                             <1>       ; 18/03/2016, 24/03/2016
                             <1>       ; 11/03/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>       ; 30/07/2011 (DRV_FAT.ASM)
                             <1>       ; 11/09/2010
                             <1>       ; INPUT ->
                             <1>       ;      ESI = Logical dos drv desc. table address
                             <1>       ;      EAX = Last cluster
                             <1>       ; OUTPUT ->
                             <1>       ;      ESI = Logical dos drv desc. table address
                             <1>       ;      EAX = New Last cluster (next cluster)
                             <1>       ;      cf = 1 -> error code in EAX (AL)
                             <1>       ;      cf = 1 -> DX = sectors per cluster
                             <1>       ;      ECX = Free sectors
                             <1>       ; NOTE:
                             <1>       ; This procedure does not update lm date&time !
                             <1>       ;
                             <1>       ; (Modified registers: EAX, EBX, ECX, EDX, EDI)
                             <1>       ;
                             <1>
0000C451 A3[F0640100]        <1>       mov    [FAT_anc_LCluster], eax
                             <1>
0000C456 E844F9FFFF          <1>       call   get_first_free_cluster
0000C45B 720B                <1>       jc     short loc_add_new_cluster_retn
                             <1>       ; EAX >= 2 and EAX < FFFFFFFFh is valid
                             <1>
0000C45D 89C2                <1>       mov    edx, eax
                             <1>
0000C45F 42                  <1>       inc    edx
                             <1>       ;jnz   short loc_add_new_cluster_check_ffc_eax
0000C460 7516                <1>       jnz    short loc_add_new_cluster_save_fcc
                             <1>
                             <1> loc_add_new_cluster_no_disk_space_retn:
0000C462 B827000000          <1>       mov    eax, 27h ; MSDOS err => insufficient disk space
                             <1> loc_add_new_cluster_stc_retn:
0000C467 F9                  <1>       stc
                             <1> loc_add_new_cluster_retn:
0000C468 0FB65E13            <1>       movzx  ebx, byte [esi+LD_BPB+SecPerClust]
0000C46C 8B4E74              <1>       mov    ecx, [esi+LD_FreeSectors]
                             <1>       ;xor   edx, edx
                             <1>       ;stc
0000C46F C3                  <1>       retn
                             <1>
                             <1> loc_anc_invalid_format_stc_retn:
```

```
0000C470 F9                  <1>       stc
                             <1> loc_add_new_cluster_invalid_format_retn:
                             <1>       ; 15/10/2016 (0Bh -> 28)
0000C471 B81C000000          <1>       mov    eax, 28 ; Invalid format
0000C476 EBF0                <1>       jmp    short loc_add_new_cluster_retn
                             <1>
                             <1> ;loc_add_new_cluster_check_ffc_eax:
                             <1> ;     cmp    eax, 2
                             <1> ;     jb     short loc_add_new_cluster_invalid_format_retn
                             <1>
                             <1> loc_add_new_cluster_save_fcc:
0000C478 A3[F4640100]        <1>       mov    [FAT_anc_FFCluster], eax
                             <1>
0000C47D 83E802              <1>       sub    eax, 2
0000C480 0FB65E13            <1>         movzx   ebx, byte [esi+LD_BPB+SecPerClust]
0000C484 F7E3                <1>       mul    ebx
0000C486 09D2                <1>       or     edx, edx
0000C488 75E6                <1>       jnz    short loc_anc_invalid_format_stc_retn
                             <1>
                             <1> loc_add_new_cluster_allocate_cluster:
                             <1>       ; 18/03/2016
0000C48A 92                  <1>       xchg   edx, eax ; eax = 0
                             <1>       ; 16/05/2016
                             <1>       ;cmp   [ClusterBuffer_Valid], al ; 0
                             <1>       ;jna   short loc_anc_clear_cluster_buffer
                             <1>       ;; 'copy' command,
                             <1>       ;; writing destination file clust after reading source file clust
                             <1>       ;mov   [ClusterBuffer_Valid], al ; 0 ; reset
                             <1>       ;jmp   short loc_add_new_cluster_write_nc_to_disk
                             <1>
                             <1> loc_anc_clear_cluster_buffer:
                             <1>       ; 11/03/2016
                             <1>       ; Clear buffer
0000C48B BF00000700          <1>       mov    edi, Cluster_Buffer ; 70000h (for current TRDOS 386 version)
0000C490 89D9                <1>       mov    ecx, ebx ; sector count
0000C492 C1E107              <1>       shl    ecx, 7 ; 1 sector = 512 bytes -> 128 double words
                             <1>       ;xor   eax, eax ; 0
0000C495 F3AB                <1>       rep    stosd
                             <1>
                             <1> loc_add_new_cluster_write_nc_to_disk:
                             <1>       ; 11/03/2016
                             <1>       ;xchg  eax, edx ; edx = 0, eax = sector offset
0000C497 89D0                <1>       mov    eax, edx
0000C499 034668              <1>         add     eax, [esi+LD_DATABegin]
0000C49C 72D3                <1>       jc     short loc_add_new_cluster_invalid_format_retn
                             <1>
0000C49E 89D9                <1>       mov    ecx, ebx ; ECX = sectors per cluster (<256)
0000C4A0 BB00000700          <1>       mov    ebx, Cluster_Buffer
0000C4A5 E81F330000          <1>       call   disk_write
0000C4AA 7307                <1>       jnc    short loc_add_new_cluster_update_fat_nlc
                             <1>
                             <1>       ; 15/10/2016 (1Dh -> 18)
0000C4AC B812000000          <1>       mov    eax, 18 ; Write Error
0000C4B1 EBB4                <1>       jmp    short loc_add_new_cluster_stc_retn
                             <1>
                             <1> loc_add_new_cluster_update_fat_nlc:
0000C4B3 A1[F4640100]        <1>       mov    eax, [FAT_anc_FFCluster]
0000C4B8 31C9                <1>       xor    ecx, ecx
0000C4BA 890D[1E610100]      <1>       mov    [FAT_ClusterCounter], ecx ; 0 ; reset
0000C4C0 49                  <1>       dec    ecx ; 0FFFFFFFFh
0000C4C1 E8ACF9FFFF          <1>       call   update_cluster
0000C4C6 7304                <1>       jnc    short loc_add_new_cluster_update_fat_plc
0000C4C8 09C0                <1>       or     eax, eax ;EAX = 0 -> cluster value is 0 or eocc
0000C4CA 759B                <1>       jnz    short loc_add_new_cluster_stc_retn
                             <1>
                             <1> loc_add_new_cluster_update_fat_plc:
0000C4CC A1[F0640100]        <1>       mov    eax, [FAT_anc_LCluster]
0000C4D1 8B0D[F4640100]      <1>       mov    ecx, [FAT_anc_FFCluster]
0000C4D7 E896F9FFFF          <1>       call   update_cluster
0000C4DC 7314                <1>       jnc    short loc_add_new_cluster_save_fat_buffer
0000C4DE 09C0                <1>       or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
0000C4E0 7410                <1>       jz     short loc_add_new_cluster_save_fat_buffer
                             <1>
                             <1> loc_anc_save_fat_buffer_err_retn:
                             <1>       ;cmp   byte [FAT_ClusterCounter], 1
                             <1>       ;jb    short loc_add_new_cluster_retn
                             <1>
0000C4E2 66BB00FF            <1>       mov    bx, 0FF00h ; recalculate free space (BL = 0)
                             <1>                       ; (BH = FFh -> Use ESI as Drv Param. Tbl.)
0000C4E6 50                  <1>       push   eax
0000C4E7 E8D8FCFFFF          <1>       call   calculate_fat_freespace
0000C4EC 58                  <1>       pop    eax
0000C4ED E975FFFFFF          <1>         jmp     loc_add_new_cluster_stc_retn
                             <1>
                             <1> loc_add_new_cluster_save_fat_buffer:
                             <1>       ;cmp   byte [FAT_BuffValidData], 2
                             <1>       ;jne   short loc_add_new_cluster_calc_FAT_freespace
                             <1>       ;Byte [FAT_BuffValidData] =  2
0000C4F2 E838FCFFFF          <1>       call   save_fat_buffer
0000C4F7 72E9                <1>       jc     short loc_anc_save_fat_buffer_err_retn
                             <1>
                             <1> loc_add_new_cluster_calc_FAT_freespace:
                             <1>       ;mov   eax, 1 ; Only one Cluster
0000C4F9 A1[1E610100]        <1>       mov    eax, [FAT_ClusterCounter]
0000C4FE 66BB01FF            <1>       mov    bx, 0FF01h ; BH = FFh -> ESI -> Dos drv desc. table
                             <1>              ; BL = 1 -> add cluster
0000C502 B301                <1>       mov    bl, 01h ; BL = 1 -> add clusters
                             <1>       ; NOTE: EAX value will be added to Free Cluster Count
                             <1>       ; (Free Cluster Count is decreased when EAX value is negative)
0000C504 E8BBFCFFFF          <1>         call    calculate_fat_freespace
                             <1>       ;ECX = 0 -> no error, ECX > 0 -> error or invalid return
0000C509 21C9                <1>       and    ecx, ecx ; ECX = 0 -> valid free sector count
0000C50B 7409                <1>       jz     short loc_add_new_cluster_return_cluster_number
```

```
                             <1>
                             <1> loc_add_new_cluster_recalc_FAT_freespace:
0000C50D 66BB00FF            <1>        mov    bx, 0FF00h  ; recalculate free space
0000C511 E8AEFCFFFF          <1>          call    calculate_fat_freespace
                             <1>        ; cf = 0
                             <1> loc_add_new_cluster_return_cluster_number:
0000C516 89C1                <1>        mov    ecx, eax ; Free sector count
0000C518 A1[F4640100]        <1>        mov    eax, [FAT_anc_FFCluster]
0000C51D 0FB65E13            <1>        movzx  ebx, byte [esi+LD_BPB+SecPerClust]
                             <1>        ;mov   edi, Cluster_Buffer
0000C521 31D2                <1>        xor    edx, edx
0000C523 C3                  <1>          retn
                             <1>
                             <1> write_cluster:
                             <1>        ; 15/10/2016
                             <1>        ; 21/03/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>        ;
                             <1>        ; INPUT ->
                             <1>        ;      EAX = Cluster Number (Sector index for SINGLIX FS)
                             <1>        ;      ESI = Logical DOS Drive Description Table address
                             <1>        ;      EBX = Cluster (File R/W) Buffer address (max. 64KB)
                             <1>        ;      Only for SINGLIX FS:
                             <1>        ;      EDX = File Number (The 1st FDT address)
                             <1>        ; OUTPUT ->
                             <1>        ;      cf = 1 -> Cluster can not be written onto disk
                             <1>        ;          EAX > 0 -> Error number
                             <1>        ;      cf = 0 -> Cluster has been written successfully
                             <1>        ;
                             <1>        ; (Modified registers: EAX, ECX, EBX, EDX)
                             <1>
0000C524 0FB64E13            <1>        movzx  ecx, byte [esi+LD_BPB+BPB_SecPerClust]
                             <1>        ; CL = 1 = [esi+LD_FS_Reserved2] ; SectPerClust for Singlix FS
                             <1>
                             <1> write_file_sectors: ; 16/03/2016
0000C528 807E0300            <1>        cmp    byte [esi+LD_FATType], 0
0000C52C 761C                <1>        jna    short write_fs_cluster
                             <1>
                             <1> write_fat_file_sectors:
0000C52E 83E802              <1>        sub    eax, 2 ; Beginning cluster number is always 2
0000C531 0FB65613            <1>        movzx  edx, byte [esi+LD_BPB+BPB_SecPerClust] ; 18/03/2016
0000C535 F7E2                <1>        mul    edx
0000C537 034668              <1>        add    eax, [esi+LD_DATABegin] ; absolute address of the cluster
                             <1>
                             <1>        ; EAX = Disk sector address
                             <1>        ; ECX = Sector count
                             <1>        ; EBX = Buffer address
                             <1>        ; (EDX = 0)
                             <1>        ; ESI = Logical DOS drive description table address
                             <1>
0000C53A E88A320000          <1>        call   disk_write
0000C53F 7306                <1>        jnc    short wclust_retn
                             <1>
                             <1>        ; 15/10/2016 (1Dh -> 18)
0000C541 B812000000          <1>        mov    eax, 18 ; Drive not ready or write error !
0000C546 C3                  <1>        retn
                             <1>
                             <1> wclust_retn:
0000C547 29C0                <1>        sub    eax, eax ; 0
0000C549 C3                  <1>        retn
                             <1>
                             <1> write_fs_cluster:
                             <1>        ; 21/03/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>        ; Singlix FS
                             <1>
                             <1>        ; EAX = Cluster number is sector index number of the file (eax)
                             <1>
                             <1>        ; EDX = File number is the first File Descriptor Table address
                             <1>        ;      of the file. (Absolute address of the FDT).
                             <1>
                             <1>        ; eax = sector index (0 for the first sector)
                             <1>        ; edx = FDT0 address
                             <1>               ; 64 KB buffer = 128 sectors (limit)
0000C54A B980000000          <1>        mov    ecx, 128 ; maximum count of sectors (before eof)
0000C54F E801000000          <1>        call   write_fs_sectors
0000C554 C3                  <1>        retn
                             <1>
                             <1> write_fs_sectors:
                             <1>        ; 21/03/2016 (TRDOS 386 =  TRDOS v2.0)
0000C555 F9                  <1>        stc
0000C556 C3                  <1>        retn
                             <1>
                             <1> get_cluster_by_index:
                             <1>        ; 29/04/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>        ; INPUT ->
                             <1>        ;      EAX = Beginning cluster
                             <1>        ;      EDX = Sector index in disk/file section
                             <1>        ;            (Only for SINGLIX file system!)
                             <1>        ;      ECX = Cluster sequence number after the beginning cluster
                             <1>        ;      ESI = Logical DOS Drive Description Table address
                             <1>        ; OUTPUT ->
                             <1>        ;      EAX = Cluster number
                             <1>        ;      cf = 1 -> Error code in AL (EAX)
                             <1>        ;
                             <1>        ;(Modified registers: EAX, ECX, EBX, EDX)
                             <1>        ;
0000C557 807E0301            <1>        cmp    byte [esi+LD_FATType], 1
0000C55B 721E                <1>          jb      short get_fs_section_by_index
                             <1>
0000C55D 3B4E78              <1>        cmp    ecx, [esi+LD_Clusters]
0000C560 7207                <1>        jb     short gcbi_1
                             <1> gcbi_0:
0000C562 F9                  <1>        stc
0000C563 B823000000          <1>        mov    eax, 23h ; Cluster not available !
```

```
                             <1>                     ; MSDOS error code: FCB unavailable
0000C568 C3                  <1>       retn
                             <1> gcbi_1:
0000C569 51                  <1>       push   ecx
0000C56A E8D9F5FFFF          <1>       call   get_next_cluster
0000C56F 59                  <1>       pop    ecx
0000C570 7203                <1>       jc     short gcbi_3
0000C572 E2F5                <1>       loop   gcbi_1
                             <1> gcbi_2:
0000C574 C3                  <1>       retn
                             <1> gcbi_3:
0000C575 09C0                <1>       or     eax, eax
0000C577 74E9                <1>       jz     short gcbi_0
0000C579 F5                  <1>       cmc    ; stc
0000C57A C3                  <1>       retn
                             <1>
                             <1> get_fs_section_by_index:
                             <1>       ; 29/04/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>       ; INPUT ->
                             <1>       ;      EAX = Beginning FDT number/address
                             <1>       ;      EDX = Sector index in disk/file section
                             <1>       ;      ECX = Sector sequence number after the beginning FDT
                             <1>       ;      ESI = Logical DOS Drive Description Table address
                             <1>       ; OUTPUT ->
                             <1>       ;      EAX = FDT number/address
                             <1>       ;      EDX = Sector index of the section (0,1,2,3,4...)
                             <1>       ;      cf = 1 -> Error code in AL (EAX)
                             <1>       ;
                             <1>       ;(Modified registers: EAX, ECX, EBX, EDX)
                             <1>       ;
0000C57B B8FFFFFFFF          <1>       mov    eax, 0FFFFFFFFh
0000C580 C3                  <1>       retn
                             <1>
                             <1> get_last_section:
                             <1>       ; 22/10/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>       ; INPUT ->
                             <1>       ;      EAX = (The 1st) FDT number/address
                             <1>       ;      ESI = Logical DOS Drive Description Table address
                             <1>       ; OUTPUT ->
                             <1>       ;      EAX = FDT number/address of the last section
                             <1>       ;      EDX = Last sector of the section (0,1,2,3,4...)
                             <1>       ;      [glc_index] = sector index number of the last sector
                             <1>       ;                  (for file, not for the last section)
                             <1>       ;
                             <1>       ;      cf = 1 -> Error code in AL (EAX)
                             <1>       ;
                             <1>       ;(Modified registers: EAX, ECX, EBX, EDX)
                             <1>       ;
0000C581 B800000000          <1>       mov    eax, 0
0000C586 BA00000000          <1>       mov    edx, 0
0000C58B C3                  <1>       retn
                                 %include 'trdosk6.s' ; 24/01/2016
                             <1> ; ****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - MAIN PROGRAM : trdosk6.s
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 31/12/2017
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 24/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                             <1> ; u1.s (27/17/2015), u2.s (03/01/2016)
                             <1> ; ****************************************************************************
                             <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
                             <1> ; TRDOS2.ASM (09/11/2011)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; INT_21H.ASM (c) 2009-2011 Erdogan TAN  [14/11/2009] Last Update: 08/11/2011
                             <1>
                             <1> sysent: ; < enter to system call >
                             <1>       ; 17/03/2017
                             <1>       ; 03/03/2017
                             <1>       ; 19/02/2017
                             <1>       ; 13/01/2017
                             <1>       ; 06/06/2016
                             <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 16/04/2015 - 19/10/2015 (Retro UNIX 386 v1)
                             <1>       ; 10/04/2013 - 18/01/2014 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'unkni' or 'sysent' is sytem entry from various traps.
                             <1>       ; The trap type is determined and an indirect jump is made to
                             <1>       ; the appropriate system call handler. If there is a trap inside
                             <1>       ; the system a jump to panic is made. All user registers are saved
                             <1>       ; and u.sp points to the end of the users stack. The sys (trap)
                             <1>       ; instructor is decoded to get the the system code part (see
                             <1>       ; trap instruction in the PDP-11 handbook) and from this
                             <1>       ; the indirect jump address is calculated. If a bad system call is
                             <1>       ; made, i.e., the limits of the jump table are exceeded, 'badsys'
                             <1>       ; is called. If the call is legitimate control passes to the
                             <1>       ; appropriate system routine.
                             <1>       ;
                             <1>       ; Calling sequence:
                             <1>       ;      Through a trap caused by any sys call outside the system.
                             <1>       ; Arguments:
                             <1>       ;      Arguments of particular system call.
                             <1>       ; ..............................................................
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 modification:
                             <1>       ;       System call number is in EAX register.
                             <1>       ;
                             <1>       ;       Other parameters are in EDX, EBX, ECX, ESI, EDI, EBP
                             <1>       ;      registers depending of function details.
                             <1>       ;
                             <1>       ; 16/04/2015
0000C58C 368925[5C030300]    <1>         mov     [ss:u.sp], esp ; Kernel stack points to return address
                             <1> 
                             <1>       ; save user registers
0000C593 1E                  <1>       push   ds
0000C594 06                  <1>       push   es
0000C595 0FA0                <1>       push   fs
0000C597 0FA8                <1>       push   gs
0000C599 60                  <1>       pushad  ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
                             <1>       ;
                             <1>       ; ESPACE = [ss:u.sp] - esp ; 4*12 = 48 ; 17/09/2015 ; 06/06/2016
                             <1>       ;      (ESPACE is size of space in kernel stack
                             <1>       ;      for saving/restoring user registers.)
                             <1>       ;
0000C59A 50                  <1>       push   eax ; 01/07/2015
0000C59B 66B81000            <1>       mov     ax, KDATA
0000C59F 8ED8                <1>         mov     ds, ax
0000C5A1 8EC0                <1>         mov     es, ax
0000C5A3 8EE0                <1>         mov     fs, ax
0000C5A5 8EE8                <1>         mov     gs, ax
0000C5A7 A1[38580100]        <1>       mov    eax, [k_page_dir]
0000C5AC 0F22D8              <1>       mov    cr3, eax
0000C5AF 58                  <1>       pop    eax ; 01/07/2015
                             <1>       ; 19/10/2015
0000C5B0 FC                  <1>       cld
                             <1>       ;
0000C5B1 FE05[5B030300]      <1>       inc    byte [sysflg]
                             <1>              ; incb sysflg / indicate a system routine is in progress
0000C5B7 FB                  <1>         sti  ; 18/01/2014
0000C5B8 0F85F39DFFFF        <1>       jnz     panic ; 24/05/2013
                             <1>              ; beq 1f
                             <1>              ; jmp panic ; / called if trap inside system
                             <1> ;1:
                             <1>       ; 17/03/2017
0000C5BE 80642438FE          <1>       and    byte [esp+ESPACE+8], ~1 ; clear carry flag
                             <1> 
                             <1>       ; 16/04/2015
0000C5C3 A3[64030300]        <1>       mov    [u.r0], eax
0000C5C8 8925[60030300]      <1>       mov    [u.usp], esp ; kernel stack points to user's registers
                             <1> 
                             <1>       ; 13/01/2017 (TRDOS 386 Feaure only !)
0000C5CE 803D[D4030300]00    <1>       cmp    byte [u.t_lock], 0 ; timer interrupt lock ?
0000C5D5 0F879D010000        <1>       ja     sysrele                ; yes, sys release only !!!
                             <1> 
                             <1>              ; mov $s.syst+2,clockp
                             <1>              ; mov r0,-(sp) / save user registers
                             <1>              ; mov sp,u.r0 / pointer to bottom of users stack
                             <1>                        ; / in u.r0
                             <1>              ; mov r1,-(sp)
                             <1>              ; mov r2,-(sp)
                             <1>              ; mov r3,-(sp)
                             <1>              ; mov r4,-(sp)
                             <1>              ; mov r5,-(sp)
                             <1>              ; mov ac,-(sp) / "accumulator" register for extended
                             <1>                           ; / arithmetic unit
                             <1>              ; mov mq,-(sp) / "multiplier quotient" register for the
                             <1>                           ; / extended arithmetic unit
                             <1>              ; mov sc,-(sp) / "step count" register for the extended
                             <1>                           ; / arithmetic unit
                             <1>              ; mov sp,u.sp / u.sp points to top of users stack
                             <1>              ; mov 18.(sp),r0 / store pc in r0
                             <1>              ; mov -(r0),r0 / sys inst in r0      10400xxx
                             <1>              ; sub $sys,r0 / get xxx code
0000C5DB C1E002              <1>       shl    eax, 2
                             <1>              ; asl r0 / multiply by 2 to jump indirect in bytes
0000C5DE 3DB8000000          <1>       cmp    eax, end_of_syscalls - syscalls
                             <1>              ; cmp r0,$2f-1f / limit of table (35) exceeded
                             <1>       ;jnb   short badsys
                             <1>              ; bhis badsys / yes, bad system call
0000C5E3 F5                  <1>       cmc
0000C5E4 9C                  <1>       pushf
0000C5E5 50                  <1>       push   eax
0000C5E6 8B2D[5C030300]      <1>       mov    ebp, [u.sp] ; Kernel stack at the beginning of sys call
0000C5EC B0FE                <1>       mov    al, 0FEh ; 11111110b
0000C5EE 1400                <1>       adc    al, 0 ; al = al + cf
0000C5F0 204508              <1>       and    [ebp+8], al ; flags (reset carry flag)
                             <1>              ; bic $341,20.(sp) / set users processor priority to 0
                             <1>                            ; / and clear carry bit
0000C5F3 5D                  <1>       pop    ebp ; eax
0000C5F4 9D                  <1>       popf
0000C5F5 0F8208020000        <1>         jc      badsys
0000C5FB A1[64030300]        <1>       mov    eax, [u.r0]
                             <1>       ; system call registers: EAX, EDX, ECX, EBX, ESI, EDI
0000C600 FFA5[06C60000]      <1>       jmp    dword [ebp+syscalls]
                             <1>              ; jmp *1f(r0) / jump indirect thru table of addresses
                             <1>                          ; / to proper system routine.
                             <1> syscalls: ; 1:
                             <1>       ; 31/12/2017
                             <1>       ; 28/02/2017
                             <1>       ; 20/02/2017
                             <1>       ; 19/02/2017
                             <1>       ; 15/10/2016
                             <1>       ; 20/05/2016
                             <1>       ; 19/05/2016
                             <1>       ; 16/05/2016
                             <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 21/09/2015
                             <1>       ; 01/07/2015
                             <1>       ; 16/04/2015 (32 bit address modification)
0000C606 [BDE60000]          <1>       dd sysver    ; 0 ; Get TRDOS 386 version number (v2.0)
0000C60A [65C80000]          <1>       dd sysexit   ; 1
0000C60E [3ACA0000]          <1>       dd sysfork   ; 2
0000C612 [6DCE0000]          <1>       dd sysread   ; 3
```

```
0000C616 [8CCE0000]          <1>       dd syswrite  ; 4
0000C61A [23CC0000]          <1>       dd sysopen   ; 5
0000C61E [44CE0000]          <1>       dd sysclose  ; 6
0000C622 [BCC90000]          <1>       dd syswait   ; 7
0000C626 [52CB0000]          <1>       dd syscreat  ; 8
0000C62A [11F50000]          <1>       dd sysrename ; 9  ; TRDOS 386, Rename File (31/12/2017)
0000C62E [8CF00000]          <1>       dd sysdelete ; 10 ; TRDOS 386, Delete File (29/12/2017)
0000C632 [A0DA0000]          <1>       dd sysexec   ; 11
0000C636 [B6F10000]          <1>       dd syschdir  ; 12
0000C63A [7BF30000]          <1>       dd systime   ; 13 ; TRDOS 386, Get Sys Date&Time (30/12/2017)
0000C63E [06CE0000]          <1>       dd sysmkdir  ; 14
0000C642 [EAF10000]          <1>       dd syschmod  ; 15 ; TRDOS 386, Change Attributes (30/12/2017)
0000C646 [F3F00000]          <1>       dd sysrmdir  ; 16 ; TRDOS 386, Remove Directory (29/12/2017)
0000C64A [7BDD0000]          <1>       dd sysbreak  ; 17
0000C64E [D0F20000]          <1>       dd sysdrive  ; 18 ; TRDOS 386, Get/Set Current Drv (30/12/2017)
0000C652 [BCDD0000]          <1>       dd sysseek   ; 19
0000C656 [CEDD0000]          <1>       dd systell   ; 20
0000C65A [32F60000]          <1>       dd sysmem    ; 21 ; TRDOS 386, Get Total&Free Mem (31/12/2017)
0000C65E [68F60000]          <1>       dd sysprompt ; 22 ; TRDOS 386, Change Cmd Prompt (31/12/2017)
0000C662 [AAF60000]          <1>       dd syspath   ; 23 ; TRDOS 386, Get/Set Run Path (31/12/2017)
0000C666 [17F70000]          <1>       dd sysenv    ; 24 ; TRDOS 386, Get/Set Env Vars (31/12/2017)
0000C66A [FCF30000]          <1>       dd sysstime  ; 25 ; TRDOS 386, Set Sys Date&Time (30/12/2017)
0000C66E [34DE0000]          <1>       dd sysquit   ; 26
0000C672 [28DE0000]          <1>       dd sysintr   ; 27
0000C676 [1FF30000]          <1>       dd sysdir    ; 28 ; TRDOS 386, Get Curr Drive&Dir (30/12/2017)
0000C67A [23CF0000]          <1>       dd sysemt    ; 29
0000C67E [5AF30000]          <1>       dd sysldrvt  ; 30 ; TRDOS 386, Get Logical DOS DDT (30/12/2017)
0000C682 [D4D00000]          <1>       dd sysvideo  ; 31 ; TRDOS 386 Video Functions (16/05/2016)
0000C686 [E9000100]          <1>       dd sysaudio  ; 32 ; TRDOS 386 Audio Functions (16/05/2016)
0000C68A [3CCF0000]          <1>       dd systimer  ; 33 ; TRDOS 386 Timer Functions (18/05/2016)
0000C68E [75DE0000]          <1>       dd syssleep  ; 34 ; Retro UNIX 8086 v1 feature only !
                             <1>                         ; 11/06/2014
0000C692 [A4DE0000]          <1>       dd sysmsg    ; 35 ; Retro UNIX 386 v1 feature only !
                             <1>                         ; 01/07/2015
0000C696 [7BDF0000]          <1>       dd sysgeterr ; 36 ; Retro UNIX 386 v1 feature only !
                             <1>                         ; 21/09/2015 - get last error number
0000C69A [63F00000]          <1>       dd sysfpstat ; 37 ; TRDOS 386 FPU state option (28/02/2017)
0000C69E [CCE60000]          <1>       dd syspri    ; 38 ; change priority - TRDOS 386 (20/05/2016)
0000C6A2 [78C70000]          <1>       dd sysrele   ; 39 ; TRDOS 386 (19/05/2016) (0 -> 39)
0000C6A6 [FFE70000]          <1>       dd sysfff    ; 40 ; Find First File - TRDOS 386 (15/10/2016)
0000C6AA [DEE80000]          <1>       dd sysfnf    ; 41 ; Find Next File - TRDOS 386 (15/10/2016)
0000C6AE [4EEF0000]          <1>       dd sysalloc  ; 42 ; Allocate contiguous memory block/pages
                             <1>                         ; TRDOS 386 (19/02/2017) DMA buff fuctions
0000C6B2 [0CF00000]          <1>       dd sysdalloc ; 43 ; Deallocate contiguous memory block/pages
                             <1>                         ; TRDOS 386 (19/02/2017) DMA buff fuctions
0000C6B6 [47F00000]          <1>       dd syscalbac ; 44 ; IRQ Callback and Signal Response Byte
                             <1>                         ; service setup - TRDOS 386 (20/02/2017)
                             <1>                         ; 28/08/2017 (20/08/2017)
0000C6BA [6D090100]          <1>       dd sysdma    ; 45 ; TRDOS 386 - (ISA) DMA service
                             <1>
                             <1> end_of_syscalls:
                             <1>
                             <1> error:
                             <1>       ; 18/05/2016
                             <1>       ; 13/05/2016
                             <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 16/04/2015 - 17/09/2015 (Retro UNIX 386 v1)
                             <1>       ; 10/04/2013 - 07/08/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'error' merely sets the error bit off the processor status (c-bit)
                             <1>       ; then falls right into the 'sysret', 'sysrele' return sequence.
                             <1>       ;
                             <1>       ; INPUTS -> none
                             <1>       ; OUTPUTS ->
                             <1>       ;      processor status - carry (c) bit is set (means error)
                             <1>       ;
                             <1>       ; 26/05/2013 (Stack pointer must be reset here!
                             <1>       ;            Because, jumps to error procedure
                             <1>       ;            disrupts push-pop nesting balance)
                             <1>       ;
0000C6BE 8B2D[5C030300]      <1>       mov    ebp, [u.sp] ; interrupt (system call) return (iretd) address
0000C6C4 804D0801            <1>       or     byte [ebp+8], 1  ; set carry bit of flags register
                             <1>                           ; (system call will return with cf = 1)
                             <1>             ; bis $1,20.(r1) / set c bit in processor status word below
                             <1>                            ; / users stack
                             <1>       ; 17/09/2015
0000C6C8 83ED30              <1>       sub    ebp, ESPACE ; 48 ; total size of stack frame ('sysdefs.inc')
                             <1>                           ; for saving/restoring user registers
                             <1>       ;cmp   ebp, [u.usp]
                             <1>       ;je    short err0
0000C6CB 892D[60030300]      <1>       mov    [u.usp], ebp
                             <1> ;err0:
                             <1>       ; 01/09/2015
0000C6D1 8B25[60030300]      <1>       mov    esp, [u.usp]     ; Retro Unix 8086 v1 modification!
                             <1>                              ; 10/04/2013
                             <1>                              ; (If an I/O error occurs during disk I/O,
                             <1>                              ; related procedures will jump to 'error'
                             <1>                              ; procedure directly without returning to
                             <1>                              ; the caller procedure. So, stack pointer
                             <1>                                     ; must be restored here.)
                             <1>       ; 13/05/2016
                             <1>       ; NOTE: (The last) error code is in 'u.error', it can be retrieved by
                             <1>       ;      'get last error' system call later.
                             <1>
                             <1>       ; 03/09/2015 - 09/06/2015 - 07/08/2013
0000C6D7 C605[C6030300]00    <1>       mov    byte [u.kcall], 0 ; namei_r, mkdir_w reset
                             <1>
                             <1> sysret: ; < return from system call>
                             <1>       ; 01/03/2017
                             <1>       ; 28/02/2017
                             <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 16/04/2015 - 10/09/2015 (Retro UNIX 386 v1)
                             <1>       ; 10/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'sysret' first checks to see if process is about to be
                             <1>       ; terminated (u.bsys). If it is, 'sysexit' is called.
                             <1>       ; If not, following happens:
                             <1>       ;      1) The user's stack pointer is restored.
                             <1>       ;      2) r1=0 and 'iget' is called to see if last mentioned
                             <1>       ;         i-node has been modified. If it has, it is written out
                             <1>       ;         via 'ppoke'.
                             <1>       ;      3) If the super block has been modified, it is written out
                             <1>       ;         via 'ppoke'.
                             <1>       ;      4) If the dismountable file system's super block has been
                             <1>       ;         modified, it is written out to the specified device
                             <1>       ;         via 'ppoke'.
                             <1>       ;      5) A check is made if user's time quantum (uquant) ran out
                             <1>       ;         during his execution. If so, 'tswap' is called to give
                             <1>       ;         another user a chance to run.
                             <1>       ;      6) 'sysret' now goes into 'sysrele'.
                             <1>       ;          (See 'sysrele' for conclusion.)
                             <1>       ;
                             <1>       ; Calling sequence:
                             <1>       ;      jump table or 'br sysret'
                             <1>       ; Arguments:
                             <1>       ;      -
                             <1>       ; ...............................................................
                             <1>       ;
                             <1>       ; ((AX=r1 for 'iget' input))
                             <1>       ;
0000C6DE 31C0                <1>       xor    eax, eax ; 28/02/2017
                             <1> sysret0: ; 29/07/2015 (eax = 0, jump from sysexec)
0000C6E0 FEC0                <1>       inc    al ; 04/05/2013
0000C6E2 3805[B2030300]      <1>       cmp    [u.bsys], al ; 1
                             <1>              ; tstb u.bsys / is a process about to be terminated because
0000C6E8 0F8377010000        <1>         jnb     sysexit ; 04/05/2013
                             <1>              ; bne sysexit / of an error? yes, go to sysexit
                             <1>       ;mov   esp, [u.usp] ; 24/05/2013 (that is not needed here)
                             <1>              ; mov u.sp,sp / no point stack to users stack
0000C6EE FEC8                <1>       dec    al ; mov ax, 0
                             <1>              ; clr r1 / zero r1 to check last mentioned i-node
0000C6F0 E8CD300000          <1>       call   iget
                             <1>              ; jsr r0,iget / if last mentioned i-node has been modified
                             <1>                          ; / it is written out
                             <1>       ; 10/01/2017
                             <1>       ; 09/01/2017
                             <1> ;sysrele: ; < release >
                             <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 16/04/2015 - 14/10/2015 (Retro UNIX 386 v1)
                             <1>       ; 10/04/2013 - 07/03/2014 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'sysrele' first calls 'tswap' if the time quantum for a user is
                             <1>       ;  zero (see 'sysret'). It then restores the user's registers and
                             <1>       ; turns off the system flag. It then checked to see if there is
                             <1>       ; an interrupt from the user by calling 'isintr'. If there is,
                             <1>       ; the output gets flashed (see isintr) and interrupt action is
                             <1>       ; taken by a branch to 'intract'. If there is no interrupt from
                             <1>       ; the user, a rti is made.
                             <1>       ;
                             <1>       ; Calling sequence:
                             <1>       ;      Fall through a 'bne' in 'sysret' & ?
                             <1>       ; Arguments:
                             <1>       ;      -
                             <1>       ; ...............................................................
                             <1>       ;
                             <1>       ; 23/02/2014 (swapret)
                             <1>       ; 22/09/2013
                             <1> sysrel0: ;1:
0000C6F5 803D[A8030300]00    <1>       cmp    byte [u.quant], 0 ; 16/05/2013
                             <1>              ; tstb uquant / is the time quantum 0?
0000C6FC 7705                <1>         ja      short swapret
                             <1>              ; bne 1f / no, don't swap it out
                             <1> sysrelease: ; 07/12/2013 (jump from 'clock')
0000C6FE E8821E0000          <1>       call   tswap
                             <1>              ; jsr r0,tswap / yes, swap it out
                             <1>
                             <1> ; Retro Unix 8086 v1 feature: return from 'swap' to 'swapret' address.
                             <1> swapret: ;1:
                             <1>       ; 10/09/2015
                             <1>       ; 01/09/2015
                             <1>       ; 14/05/2015
                             <1>       ; 16/04/2015 (Retro UNIX 386 v1 - 32 bit, pm modifications)
                             <1>       ; 26/05/2013 (Retro UNIX 8086 v1)
                             <1>       ; cli
                             <1>       ; 24/07/2015
                             <1>       ;
                             <1>       ;; 'esp' must be already equal to '[u.usp]' here !
                             <1>       ;; mov esp, [u.usp]
                             <1>
                             <1>       ; 22/09/2013
0000C703 E8BB300000          <1>       call   isintr
                             <1>       ; 20/10/2013
0000C708 7405                <1>       jz     short sysrel1
0000C70A E83F010000          <1>       call   intract
                             <1>              ; jsr r0,isintr / is there an interrupt from the user
                             <1>              ;     br intract / yes, output gets flushed, take interrupt
                             <1>                             ; / action
                             <1> sysrel1:
0000C70F FA                  <1>       cli    ; 14/10/2015
                             <1> sysrel2:
                             <1>       ; 28/02/2017
                             <1>       ; Check if there is a (delayed) callback for current user/process
0000C710 A0[D7030300]        <1>       mov    al, [u.irqwait]
0000C715 240F                <1>       and    al, 0Fh ; is there a waiting IRQ callback service ?
0000C717 7444                <1>       jz     short sysrel8 ; no
                             <1>
                             <1>       ; Set return to IRQ callback service and return from the service
0000C719 0FB6D8              <1>       movzx ebx, al
0000C71C 883D[D7030300]      <1>       mov    [u.irqwait], bh ; 0 ; reset
0000C722 8A9B[08160100]      <1>       mov    bl, [ebx+IRQenum] ; (available) IRQ index +1 (1 to 9)
                             <1>       ; 01/03/2017
0000C728 FECB                <1>       dec    bl ; IRQ index number, 0 to 8
0000C72A 7831                <1>       js     short sysrel8 ; 0 -> FFh (not in use!?)
                             <1>       ;
0000C72C A0[B3030300]        <1>       mov    al, [u.uno] ; current process (user) number
0000C731 3883[6E6B0100]      <1>       cmp    [ebx+IRQ.owner], al
0000C737 7524                <1>       jne    short sysrel8 ; it is not the current user/process !?
0000C739 F683[806B0100]01    <1>       test   byte [ebx+IRQ.method], 1 ; callback ?
0000C740 741B                <1>       jz     short sysrel8 ; not a callback method !?
                             <1>
0000C742 8B93[926B0100]      <1>       mov    edx, [ebx+IRQ.addr] ; IRQ callback service address (virtual)
0000C748 C605[D8030300]01    <1>       mov    byte [u.r_lock], 1 ; IRQ callback service in progress flag
                             <1>
0000C74F E8D91E0000          <1>       call   wswap ; save user's registers & status
                             <1>                    ;      (for return from IRQ callback service)
                             <1>
0000C754 8B2D[5C030300]      <1>       mov    ebp, [u.sp]; kernel's stack, points to EIP (user)
0000C75A 895500              <1>       mov    [ebp], edx ; IRQ call back service address
                             <1> sysrel8:
0000C75D FE0D[5B030300]      <1>       dec    byte [sysflg]
                             <1>              ; decb sysflg / turn system flag off
                             <1>
0000C763 A1[B8030300]        <1>       mov    eax, [u.pgdir]
0000C768 0F22D8              <1>       mov    cr3, eax  ; 1st PDE points to Kernel Page Table 0 (1st 4 MB)
                             <1>                       ; (others are different than kernel page tables)
                             <1>       ; 10/09/2015
0000C76B 61                  <1>       popad ; edi, esi, ebp, temp (icrement esp by 4), ebx, edx, ecx, eax
                             <1>              ; mov (sp)+,sc / restore user registers
                             <1>              ; mov (sp)+,mq
                             <1>              ; mov (sp)+,ac
                             <1>              ; mov (sp)+,r5
                             <1>              ; mov (sp)+,r4
                             <1>              ; mov (sp)+,r3
                             <1>              ; mov (sp)+,r2
                             <1>       ;
0000C76C A1[64030300]        <1>       mov    eax, [u.r0]  ; ((return value in EAX))
0000C771 0FA9                <1>       pop    gs
0000C773 0FA1                <1>       pop    fs
0000C775 07                  <1>       pop    es
0000C776 1F                  <1>       pop    ds
                             <1>       ;or    word [esp+8], 200h ; 22/01/2017 ; force enabling interrupts
0000C777 CF                  <1>       iretd
                             <1>              ; rti / no, return from interrupt
                             <1>
                             <1> sysrele:
                             <1>       ; 24/03/2017
                             <1>       ; 28/02/2017
                             <1>       ; 27/02/2017
                             <1>       ; 29/01/2017
                             <1>       ; 14/01/2017
                             <1>       ; 13/01/2017
                             <1>       ; 09/01/2017, 10/01/2017, 12/01/2017
                             <1>       ; Major modification for TRDOS 386 (CallBack return)
                             <1>       ;
                             <1>       ; 'sysrele' system call restores previously saved
                             <1>       ; registers and addresses of the process
                             <1>       ; (Main purpose -in TRDOS 386- is to return from
                             <1>       ; timer callback service routine in ring 3 -user mode-.)
                             <1>       ;
                             <1>       ; check if the process is in timer callback phase
0000C778 803D[D4030300]00    <1>       cmp    byte [u.t_lock], 0 ; TIMER INT LOCK
                             <1>       ;je    short sysrel0 ; classic (Retro UNIX 386 type) sysrele
0000C77F 7734                <1>       ja     short sysrel3
                             <1>       ; 27/02/2017
0000C781 803D[D8030300]00    <1>       cmp    byte [u.r_lock], 0 ; IRQ callback lock
0000C788 0F8667FFFFFF        <1>       jna    sysrel0 ; classic sysrele ; 24/03/2017
0000C78E E859000000          <1>       call   sysrel7
0000C793 803D[D8030300]00    <1>       cmp    byte [u.r_lock], 0 ; IRQ callback service lock
0000C79A 7628                <1>       jna    short sysrel4
0000C79C C605[D8030300]00    <1>       mov    byte [u.r_lock], 0 ; reset
                             <1>       ;mov   byte [u.irqwait], 0 ; reset ; 28/02/2017
0000C7A3 A0[D9030300]        <1>       mov    al, [u.r_mode]
0000C7A8 08C0                <1>       or     al, al
0000C7AA 7518                <1>       jnz    short sysrel4
0000C7AC FEC8                <1>       dec    al
0000C7AE A2[D9030300]        <1>       mov    [u.r_mode], al ; 0FFh ; not necessary !?
0000C7B3 EB32                <1>       jmp    short sysrel6
                             <1> sysrel3:
                             <1>       ; 27/02/2017
0000C7B5 E832000000          <1>       call   sysrel7
                             <1>       ; 14/01/2017
0000C7BA 28C0                <1>       sub    al, al
0000C7BC 3805[D4030300]      <1>       cmp    [u.t_lock], al ; 0 ; TIMER INT LOCK
0000C7C2 770E                <1>       ja     short sysrel5 ; yes
                             <1> sysrel4:
                             <1>       ; 29/01/2017
0000C7C4 8B44241C            <1>       mov    eax, [esp+28] ; eax
0000C7C8 A3[64030300]        <1>       mov    [u.r0], eax
0000C7CD E93EFFFFFF          <1>       jmp    sysrel2
                             <1> sysrel5:
0000C7D2 A2[D4030300]        <1>       mov    [u.t_lock], al ; 0 ; reset
0000C7D7 A0[D5030300]        <1>       mov    al, [u.t_mode]
0000C7DC 20C0                <1>       and    al, al
                             <1>       ;jnz   short sysrel2 ; 0FFh ; user mode
0000C7DE 75E4                <1>       jnz    short sysrel4 ; 29/01/2017
0000C7E0 FEC8                <1>       dec    al
0000C7E2 A2[D5030300]        <1>       mov    [u.t_mode], al ; 0FFh ; not necessary !?
                             <1> sysrel6:
                             <1>       ; cpu will continue from the interrupted sytem call addr
```

```
0000C7E7 61                  <1>       popad        ; edi, esi, ebp, esp, ebx, edx, ecx, eax
0000C7E8 83C410              <1>       add    esp, 16      ; pass segment segisters: ds, es, fs, gs
0000C7EB CF                  <1>       iretd        ; eip, cs, eflags
                             <1>
                             <1> sysrel7:
0000C7EC 0FB61D[B3030300]    <1>       movzx  ebx, byte [u.uno] ; current process number
0000C7F3 66C1E302            <1>       shl    bx, 2
                             <1>       ;cmp   [ebx+p.tcb-4], eax ; 0 ; is there callback address ?
                             <1>       ;jna   short sysrel0
                             <1>       ; yes, reset callback address then restore process registers
                             <1>       ;mov   [ebx+p.tcb-4], eax ; 0 ; reset
0000C7F7 8B83[BC000300]      <1>       mov     eax, [ebx+p.upage-4] ; UPAGE address
0000C7FD FA                  <1>       cli    ; disable interrupts till 'iretd'
0000C7FE E9621E0000          <1>       jmp    rswap ; restore process 'u' structure
                             <1>
                             <1> badsys:
                             <1>       ; 25/12/2016
                             <1>       ; 18/04/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 17/04/2011 (TRDOS v1.0, 'IFC.ASM')
                             <1>       ; 03/02/2011 ('trdos_ifc_routine')
                             <1>       ;
                             <1>       ; 16/04/2015 (Retro UNIX 386 v1, 'badsys')
                             <1>       ; (EIP, EAX values will be shown on screen with error message)
                             <1>       ; (EIP = 'CD 40h' instruction address -INT 40h-)
                             <1>       ; (EAX = Function number)
                             <1>       ;
0000C803 FE05[B2030300]      <1>       inc    byte [u.bsys]
                             <1>       ;
0000C809 8B1D[5C030300]      <1>       mov    ebx, [u.sp] ; esp at the beginning of 'sysent'
0000C80F 8B03                <1>       mov    eax, [ebx] ; EIP (return address, not 'INT 30h' address)
0000C811 83E802              <1>       sub    eax, 2 ; CDh, ##h
0000C814 E8F06AFFFF          <1>       call   dwordtohex
0000C819 8915[E1130100]      <1>       mov    [eip_str], edx
0000C81F A3[E5130100]        <1>       mov    [eip_str+4], eax
0000C824 A1[64030300]        <1>       mov    eax, [u.r0]
0000C829 E8DB6AFFFF          <1>       call   dwordtohex
0000C82E 8915[D0130100]      <1>       mov    [eax_str], edx
0000C834 A3[D4130100]        <1>       mov    [eax_str+4], eax
                             <1>
0000C839 66C705[C5130100]34- <1>       mov    word [int_num_str], SYSCALL_INT_NUM ; 25/12/2016
0000C841 30                  <1>
                             <1>
0000C842 BE[97130100]        <1>       mov    esi, ifc_msg ; "invalid funtion call !" msg (trdosk9.s)
0000C847 E8119BFFFF          <1>       call   print_msg
                             <1>
0000C84C EB17                <1>       jmp    sysexit
                             <1>
                             <1> intract: ; / interrupt action
                             <1>       ; 14/10/2015
                             <1>       ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 09/05/2013 - 07/12/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 modification !
                             <1>       ; (Process/task switching and quit routine by using
                             <1>       ; Retro UNIX 8086 v1 keyboard interrupt output.))
                             <1>       ;
                             <1>       ; input -> 'u.quit'  (also value of 'u.intr' > 0)
                             <1>       ; output -> If value of 'u.quit' = FFFFh ('ctrl+brk' sign)
                             <1>       ;            'intract' will jump to 'sysexit'.
                             <1>       ;          Intract will return to the caller
                             <1>       ;            if value of 'u.quit' <> FFFFh.
                             <1>       ; 14/10/2015
0000C84E FB                  <1>       sti
                             <1>       ; 07/12/2013
0000C84F 66FF05[AC030300]    <1>       inc    word [u.quit]
0000C856 7408                <1>       jz     short intrct0 ; FFFFh -> 0
0000C858 66FF0D[AC030300]    <1>       dec    word [u.quit]
                             <1>       ; 16/04/2015
0000C85F C3                  <1>       retn
                             <1> intrct0:
0000C860 58                  <1>       pop    eax ; call intract -> retn
                             <1>       ;
0000C861 31C0                <1>       xor    eax, eax
0000C863 FEC0                <1>       inc    al  ; mov ax, 1
                             <1> ;;;
                             <1>       ; UNIX v1 original 'intract' routine...
                             <1>       ; / interrupt action
                             <1>              ;cmp *(sp),$rti / are you in a clock interrupt?
                             <1>              ; bne 1f / no, 1f
                             <1>              ; cmp (sp)+,(sp)+ / pop clock pointer
                             <1>       ; 1: / now in user area
                             <1>              ; mov r1,-(sp) / save r1
                             <1>              ; mov u.ttyp,r1
                             <1>                     ; / pointer to tty buffer in control-to r1
                             <1>              ; cmpb 6(r1),$177
                             <1>                     ; / is the interrupt char equal to "del"
                             <1>              ; beq 1f / yes, 1f
                             <1>              ; clrb 6(r1)
                             <1>                      ; / no, clear the byte
                             <1>                     ; / (must be a quit character)
                             <1>              ; mov (sp)+,r1 / restore r1
                             <1>              ; clr u.quit / clear quit flag
                             <1>              ; bis $20,2(sp)
                             <1>                     ; / set trace for quit (sets t bit of
                             <1>                     ; / ps-trace trap)
                             <1>              ; rti   ;  / return from interrupt
                             <1>       ; 1: / interrupt char = del
                             <1>              ; clrb 6(r1) / clear the interrupt byte
                             <1>                       ; / in the buffer
                             <1>              ; mov (sp)+,r1 / restore r1
                             <1>              ; cmp u.intr,$core / should control be
                             <1>                           ; / transferred to loc core?
                             <1>              ; blo 1f
                             <1>             ; jmp *u.intr / user to do rti yes,
                             <1>                           ; / transfer to loc core
                             <1>       ; 1:
                             <1>             ; sys 1 / exit
                             <1> 
                             <1> sysexit: ; <terminate process>
                             <1>       ; 14/11/2017
                             <1>       ; 27/05/2017
                             <1>       ; 10/04/2017
                             <1>       ; 26/02/2017, 28/02/2017
                             <1>       ; 02/01/2017, 23/01/2017
                             <1>       ; 06/06/2016, 10/06/2016
                             <1>       ; 19/05/2016, 23/05/2016
                             <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 16/04/2015 - 01/09/2015 (Retro UNIX 386 v1)
                             <1>       ; 19/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'sysexit' terminates a process. First each file that
                             <1>       ; the process has opened is closed by 'flose'. The process
                             <1>       ; status is then set to unused. The 'p.pid' table is then
                             <1>       ; searched to find children of the dying process. If any of
                             <1>       ; children are zombies (died by not waited for), they are
                             <1>       ; set free. The 'p.pid' table is then searched to find the
                             <1>       ; dying process's parent. When the parent is found, it is
                             <1>       ; checked to see if it is free or it is a zombie. If it is
                             <1>       ; one of these, the dying process just dies. If it is waiting
                             <1>       ; for a child process to die, it notified that it doesn't
                             <1>       ; have to wait anymore by setting it's status from 2 to 1
                             <1>       ; (waiting to active). It is awakened and put on runq by
                             <1>       ; 'putlu'. The dying process enters a zombie state in which
                             <1>       ; it will never be run again but stays around until a 'wait'
                             <1>       ; is completed by it's parent process. If the parent is not
                             <1>       ; found, process just dies. This means 'swap' is called with
                             <1>       ; 'u.uno=0'. What this does is the 'wswap' is not called
                             <1>       ; to write out the process and 'rswap' reads the new process
                             <1>       ; over the one that dies..i.e., the dying process is
                             <1>       ; overwritten and destroyed.
                             <1>       ;
                             <1>       ; Calling sequence:
                             <1>       ;     sysexit or conditional branch.
                             <1>       ; Arguments:
                             <1>       ;     -
                             <1>       ; ...............................................................
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 modification:
                             <1>       ;       System call number (=1) is in EAX register.
                             <1>       ;
                             <1>       ;       Other parameters are in EDX, EBX, ECX, ESI, EDI, EBP
                             <1>       ;       registers depending of function details.
                             <1>       ;
                             <1>       ; ('swap' procedure is mostly different than original UNIX v1.)
                             <1>       ;
                             <1> ; / terminate process
                             <1>       ; AX = 1
0000C865 6648                <1>       dec    ax ; 0
0000C867 66A3[AA030300]      <1>       mov    [u.intr], ax ; 0
                             <1>             ; clr u.intr / clear interrupt control word
                             <1>             ; clr r1 / clear r1
                             <1> sysexit_0:
                             <1>       ; 23/01/2017
                             <1>       ; 02/01/2017
                             <1>       ; 10/06/2016
                             <1>       ; 06/06/2016
                             <1>       ; 23/05/2016
                             <1>       ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; Check and stop/clear timer event(s) of this (dying) process
                             <1>       ; if there is.
                             <1> 
                             <1>       ; 02/01/2017
0000C86D FA                  <1>       cli    ; disable interrupts
                             <1>       ; 23/01/2017 - reset timer frequency (to 18.2Hz)
0000C86E B036                <1>       mov    al, 00110110b ; 36h
0000C870 E643                <1>       out    43h, al
0000C872 28C0                <1>       sub    al, al ; 0
0000C874 E640                <1>       out    40h, al ; LB
0000C876 E640                <1>       out    40h, al ; HB
                             <1>       ;
0000C878 0FB61D[B3030300]    <1>       movzx  ebx, byte [u.uno]
                             <1>       ;mov   bl, [u.uno] ; process number of dying process
0000C87F 3883[FF000300]      <1>       cmp    byte [ebx+p.timer-1], al ; 0
0000C885 763A                <1>       jna    short sysexit_12 ; no timer events for this process
0000C887 8883[FF000300]      <1>       mov    byte [ebx+p.timer-1], al ; 0 ; reset
                             <1>       ;mov   al, [timer_events]
                             <1>       ;or    al, al
                             <1>       ;jz    short sysexit_12 ; no timer events
                             <1>       ;mov   cl, al
0000C88D 8A0D[CF650100]      <1>       mov    cl, [timer_events] ; 14/11/2017
                             <1>       ;cli   ; disable interrupts
0000C893 B410                <1>       mov    ah, 16 ; number of available timer events
0000C895 BE[60040300]        <1>       mov    esi, timer_set ; beginning address of timer events
                             <1> sysexit_7:
0000C89A 8A06                <1>       mov    al, [esi] ; process number (of timer event)
0000C89C 38D8                <1>       cmp    al, bl ; process number comparison
0000C89E 7411                <1>       je     short sysexit_10
0000C8A0 20C0                <1>       and    al, al
0000C8A2 7404                <1>       jz     short sysexit_9
                             <1> sysexit_8:
0000C8A4 FEC9                <1>       dec    cl
0000C8A6 7416                <1>       jz     short sysexit_11
                             <1> sysexit_9:
0000C8A8 FECC                <1>       dec    ah
0000C8AA 7415                <1>       jz     short sysexit_12
0000C8AC 83C610              <1>       add    esi, 16
```

```
0000C8AF EBE9                 <1>       jmp    short sysexit_7
                              <1> 
                              <1> sysexit_10:
                              <1>       ;mov   byte [esi], 0
0000C8B1 66C7060000           <1>       mov    word [esi], 0
                              <1>       ;mov   dword [esi+12], 0
                              <1>       ;
0000C8B6 FE0D[CF650100]       <1>       dec    byte [timer_events] ; 02/01/2017
                              <1>       ;
0000C8BC EBE6                 <1>       jmp    short sysexit_8
                              <1> 
                              <1> sysexit_11:
0000C8BE 6629C0               <1>       sub    ax, ax ; 0 ; 26/02/2017
                              <1> sysexit_12:
                              <1>       ; 26/02/2017 (Unlink IRQ callbacks belong to the user)
0000C8C1 803D[D6030300]00     <1>       cmp    byte [u.irqc], 0 ; Count of IRQ callbacks
0000C8C8 7E2E                 <1>       jng    short sysexit_16 ; zero or invalid
                              <1>       ; 28/02/2017
                              <1>       ; clear IRQ callback flags (for 'sysrele' and 'sysret')
0000C8CA A2[D7030300]         <1>       mov    [u.irqwait], al ; 0 ; force to clear waiting flag
0000C8CF A2[D8030300]         <1>       mov    [u.r_lock], al ; 0 ; force to clear busy flag
0000C8D4 BE[6E6B0100]         <1>       mov    esi, IRQ.owner
                              <1> sysexit_13:
0000C8D9 AC                   <1>       lodsb
0000C8DA 3A05[B3030300]       <1>       cmp    al, [u.uno] ; owner  = current user ?
0000C8E0 750C                 <1>       jne    short sysexit_14
0000C8E2 C646FF00             <1>       mov    byte [esi-1], 0 ; owner = 0 : Free
0000C8E6 FE0D[D6030300]       <1>       dec    byte [u.irqc]
0000C8EC 7408                 <1>       jz     short sysexit_15
                              <1> sysexit_14:
0000C8EE 81FE[766B0100]       <1>       cmp    esi, IRQ.owner + 8 ; the last IRQ index number ?
0000C8F4 76E3                 <1>       jna    short sysexit_13 ; no
                              <1> sysexit_15:
0000C8F6 30C0                 <1>       xor    al, al ; 0
                              <1> sysexit_16: ; 2:
0000C8F8 FB                   <1>       sti    ; enable interrupts
                              <1>       ;
                              <1>       ; AX = 0
                              <1> sysexit_1: ; 1:
                              <1>       ; AX = File descriptor
                              <1>              ; / r1 has file descriptor (index to u.fp list)
                              <1>              ; / Search the whole list
0000C8F9 E89A130000           <1>       call   fclose
                              <1>              ; jsr r0,fclose / close all files the process opened
                              <1>       ;; ignore error return
                              <1>              ; br .+2 / ignore error return
                              <1>       ;inc   ax
0000C8FE FEC0                 <1>       inc    al
                              <1>              ; inc r1 / increment file descriptor
                              <1>       ;cmp   ax, 10
0000C900 3C0A                 <1>       cmp    al, 10
                              <1>              ; cmp r1,$10. / end of u.fp list?
0000C902 72F5                 <1>       jb     short sysexit_1
                              <1>              ; blt 1b / no, go back
                              <1>       ;movzx ebx, byte [u.uno]
0000C904 8A1D[B3030300]       <1>       mov    bl, [u.uno] ; 02/01/2017
                              <1>              ; movb u.uno,r1 / yes, move dying process's number to r1
0000C90A 88A3[AF000300]       <1>       mov    [ebx+p.stat-1], ah ; 0, SFREE
                              <1>              ; clrb p.stat-1(r1) / free the process
                              <1>       ; 10/04/2017
0000C910 381D[E56B0100]       <1>       cmp    [audio_user], bl
0000C916 7518                 <1>       jne    short sysexit_17
                              <1>       ; reset audio device (current) owner and 'initializated' flag
0000C918 883D[E56B0100]       <1>       mov    [audio_user], bh ; 0
                              <1>       ; 27/05/2017
0000C91E 8B0D[D06B0100]       <1>       mov    ecx,  [audio_buffer]
0000C924 09C9                 <1>       or     ecx, ecx
0000C926 7408                 <1>       jz     short sysexit_17
                              <1>       ; 'deallocate_user_pages' is not necessary in sysexit !!!
                              <1>       ;push  ebx
                              <1>       ;mov   ebx, ecx
                              <1>       ;mov   ecx, [audio_buff_size]
                              <1>       ;call  deallocate_user_pages
                              <1>       ;; (Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP)
0000C928 29C9                 <1>       sub    ecx, ecx
0000C92A 890D[D06B0100]       <1>       mov    [audio_buffer], ecx ; 0
                              <1>       ;pop   ebx
                              <1> sysexit_17:
                              <1>       ;shl   bx, 1
0000C930 D0E3                 <1>       shl    bl, 1
                              <1>              ; asl r1 / use r1 for index into the below tables
0000C932 668B8B[1E000300]     <1>       mov    cx, [ebx+p.pid-2]
                              <1>              ; mov p.pid-2(r1),r3 / move dying process's name to r3
0000C939 668B93[3E000300]     <1>       mov    dx, [ebx+p.ppid-2]
                              <1>              ; mov p.ppid-2(r1),r4 / move its parents name to r4
                              <1>       ; xor  bx, bx ; 0
0000C940 30DB                 <1>       xor    bl, bl ; 0
                              <1>              ; clr r2
0000C942 31F6                 <1>       xor    esi, esi ; 0
                              <1>              ; clr r5 / initialize reg
                              <1> sysexit_2: ; 1:
                              <1>                ; / find children of this dying process,
                              <1>              ; / if they are zombies, free them
                              <1>       ;add   bx, 2
0000C944 80C302               <1>       add    bl, 2
                              <1>              ; add $2,r2 / search parent process table
                              <1>                        ; / for dying process's name
0000C947 66398B[3E000300]     <1>       cmp    [ebx+p.ppid-2], cx
                              <1>              ; cmp p.ppid-2(r2),r3 / found it?
0000C94E 7513                 <1>       jne    short sysexit_4
                              <1>              ; bne 3f / no
                              <1>       ;shr   bx, 1
0000C950 D0EB                 <1>       shr    bl, 1
```

```
                             <1>             ; asr r2 / yes, it is a parent
0000C952 80BB[AF000300]03    <1>       cmp   byte [ebx+p.stat-1], 3 ; SZOMB
                             <1>             ; cmpb p.stat-1(r2),$3 / is the child of this
                             <1>                               ; / dying process a zombie
0000C959 7506                <1>       jne   short sysexit_3
                             <1>             ; bne 2f / no
0000C95B 88A3[AF000300]      <1>       mov   [ebx+p.stat-1], ah ; 0, SFREE
                             <1>             ; clrb p.stat-1(r2) / yes, free the child process
                             <1> sysexit_3: ; 2:
                             <1>       ;shr  bx, 1
0000C961 D0E3                <1>       shl   bl, 1
                             <1>             ; asl r2
                             <1> sysexit_4: ; 3:
                             <1>             ; / search the process name table
                             <1>             ; / for the dying process's parent
0000C963 663993[1E000300]    <1>       cmp   [ebx+p.pid-2], dx
                             <1>             ; cmp p.pid-2(r2),r4 / found it?
0000C96A 7502                <1>       jne   short sysexit_5
                             <1>             ; bne 3f / no
0000C96C 89DE                <1>       mov   esi, ebx
                             <1>             ; mov r2,r5 / yes, put index to p.pid table (parents
                             <1>                       ; / process # x2) in r5
                             <1> sysexit_5: ; 3:
                             <1>       ;cmp  bx, nproc + nproc
0000C96E 80FB20              <1>       cmp   bl, nproc + nproc
                             <1>             ; cmp r2,$nproc+nproc / has whole table been searched?
0000C971 72D1                <1>       jb    short sysexit_2
                             <1>             ; blt 1b / no, go back
                             <1>             ; mov r5,r1 / yes, r1 now has parents process # x2
0000C973 21F6                <1>       and   esi, esi ; r5=r1
0000C975 7436                <1>       jz    short sysexit_6
                             <1>             ; beq 2f / no parent has been found.
                             <1>                    ; / The process just dies
0000C977 66D1EE              <1>       shr   si, 1
                             <1>             ; asr r1 / set up index to p.stat
0000C97A 8A86[AF000300]      <1>       mov   al, [esi+p.stat-1]
                             <1>             ; movb p.stat-1(r1),r2 / move status of parent to r2
0000C980 20C0                <1>       and   al, al
0000C982 7429                <1>       jz    short sysexit_6
                             <1>             ; beq 2f / if its been freed, 2f
0000C984 3C03                <1>       cmp   al, 3
                             <1>             ; cmp r2,$3 / is parent a zombie?
0000C986 7425                <1>       je    short sysexit_6
                             <1>             ; beq 2f / yes, 2f
                             <1>       ; BH = 0
0000C988 8A1D[B3030300]      <1>       mov   bl, [u.uno]
                             <1>             ; movb u.uno,r3 / move dying process's number to r3
0000C98E C683[AF000300]03    <1>       mov   byte [ebx+p.stat-1], 3  ; SZOMB
                             <1>             ; movb $3,p.stat-1(r3) / make the process a zombie
0000C995 3C01                <1>       cmp   al, 1 ; SRUN
0000C997 7414                <1>       je    short sysexit_6
                             <1>       ;cmp  al, 2
                             <1>             ; cmp r2,$2 / is the parent waiting for
                             <1>                      ; / this child to die
                             <1>       ;jne  short sysexit_6
                             <1>             ; bne 2f / yes, notify parent not to wait any more
                             <1>       ; p.stat = 2 --> waiting
                             <1>       ; p.stat = 4 --> sleeping
0000C999 C686[AF000300]01    <1>       mov   byte [esi+p.stat-1], 1 ; SRUN
                             <1>       ;dec  byte [esi+p.stat-1]
                             <1>             ; decb p.stat-1(r1) / awaken it by putting it (parent)
0000C9A0 6689F0              <1>       mov   ax, si ; r1  (process number in AL)
                             <1>       ;
                             <1>       ;mov  ebx, runq + 4
                             <1>             ; mov $runq+4,r2 / on the runq
0000C9A3 BB[54030300]        <1>       mov   ebx, runq+2 ; normal run queue ; 02/01/2017
0000C9A8 E8F01C0000          <1>       call  putlu
                             <1>             ; jsr r0, putlu
                             <1> sysexit_6:
                             <1>             ; / the process dies
0000C9AD C605[B3030300]00    <1>       mov   byte [u.uno], 0
                             <1>             ; clrb u.uno / put zero as the process number,
                             <1>                  ; / so "swap" will
0000C9B4 E8E61B0000          <1>       call  swap
                             <1>             ; jsr r0,swap / overwrite process with another process
                             <1>
                             <1> hlt_sys:
                             <1>       ;sti
                             <1> hlts0:
0000C9B9 F4                  <1>       hlt
0000C9BA EBFD                <1>       jmp   short hlts0
                             <1>             ; 0 / and thereby kill it; halt?
                             <1>
                             <1> syswait: ; < wait for a processs to die >
                             <1>       ; 17/09/2015
                             <1>       ; 02/09/2015
                             <1>       ; 01/09/2015
                             <1>       ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 24/05/2013 - 05/02/2014 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'syswait' waits for a process die.
                             <1>       ; It works in following way:
                             <1>       ;    1) From the parent process number, the parent's
                             <1>       ;      process name is found. The p.ppid table of parent
                             <1>       ;      names is then searched for this process name.
                             <1>       ;      If a match occurs, r2 contains child's process
                             <1>       ;      number. The child status is checked to see if it is
                             <1>       ;      a zombie, i.e; dead but not waited for (p.stat=3)
                             <1>       ;      If it is, the child process is freed and it's name
                             <1>       ;      is put in (u.r0). A return is then made via 'sysret'.
                             <1>       ;      If the child is not a zombie, nothing happens and
                             <1>       ;      the search goes on through the p.ppid table until
                             <1>       ;      all processes are checked or a zombie is found.
```

```
                              <1>       ;    2) If no zombies are found, a check is made to see if
                              <1>       ;      there are any children at all. If there are none,
                              <1>       ;      an error return is made. If there are, the parent's
                              <1>       ;      status is set to 2 (waiting for child to die),
                              <1>       ;      the parent is swapped out, and a branch to 'syswait'
                              <1>       ;      is made to wait on the next process.
                              <1>       ;
                              <1>       ; Calling sequence:
                              <1>       ;      ?
                              <1>       ; Arguments:
                              <1>       ;      -
                              <1>       ; Inputs: -
                              <1>       ; Outputs: if zombie found, it's name put in u.r0.
                              <1>       ; ................................................................
                              <1>       ;
                              <1>
                              <1> ; / wait for a process to die
                              <1>
                              <1> syswait_0:
0000C9BC 0FB61D[B3030300]     <1>       movzx  ebx, byte [u.uno] ; 01/09/2015
                              <1>              ; movb u.uno,r1 / put parents process number in r1
0000C9C3 D0E3                 <1>       shl    bl, 1
                              <1>       ;shl   bx, 1
                              <1>              ; asl r1 / x2 to get index into p.pid table
0000C9C5 668B83[1E000300]     <1>       mov    ax, [ebx+p.pid-2]
                              <1>              ; mov p.pid-2(r1),r1 / get the name of this process
0000C9CC 31F6                 <1>       xor    esi, esi
                              <1>              ; clr r2
0000C9CE 31C9                 <1>       xor    ecx, ecx ; 30/10/2013
                              <1>       ;xor   cl, cl
                              <1>              ; clr r3 / initialize reg 3
                              <1> syswait_1: ; 1:
0000C9D0 6683C602             <1>       add    si, 2
                              <1>              ; add $2,r2 / use r2 for index into p.ppid table
                              <1>                      ; / search table of parent processes
                              <1>                      ; / for this process name
0000C9D4 663B86[3E000300]     <1>       cmp    ax, [esi+p.ppid-2]
                              <1>              ; cmp p.ppid-2(r2),r1 / r2 will contain the childs
                              <1>                                ; / process number
0000C9DB 7535                 <1>       jne    short syswait_3
                              <1>              ;bne 3f / branch if no match of parent process name
                              <1>       ;inc   cx
0000C9DD FEC1                 <1>       inc    cl
                              <1>              ;inc r3 / yes, a match, r3 indicates number of children
0000C9DF 66D1EE               <1>       shr    si, 1
                              <1>              ; asr r2 / r2/2 to get index to p.stat table
                              <1>       ; The possible states ('p.stat' values) of a process are:
                              <1>       ;      0 = free or unused
                              <1>       ;      1 = active
                              <1>       ;      2 = waiting for a child process to die
                              <1>       ;      3 = terminated, but not yet waited for (zombie).
0000C9E2 80BE[AF000300]03     <1>       cmp    byte [esi+p.stat-1], 3 ; SZOMB, 05/02/2014
                              <1>              ; cmpb p.stat-1(r2),$3 / is the child process a zombie?
0000C9E9 7524                 <1>       jne    short syswait_2
                              <1>              ; bne 2f / no, skip it
0000C9EB 88BE[AF000300]       <1>       mov    [esi+p.stat-1], bh ; 0
                              <1>              ; clrb p.stat-1(r2) / yes, free it
0000C9F1 66D1E6               <1>       shl    si, 1
                              <1>              ; asl r2 / r2x2 to get index into p.pid table
0000C9F4 0FB786[1E000300]     <1>       movzx  eax, word [esi+p.pid-2]
0000C9FB A3[64030300]         <1>       mov    [u.r0], eax
                              <1>              ; mov p.pid-2(r2),*u.r0
                              <1>                          ; / put childs process name in (u.r0)
                              <1>       ;
                              <1>       ; Retro UNIX 386 v1 modification ! (17/09/2015)
                              <1>       ;
                              <1>       ; Parent process ID -p.ppid- field (of the child process)
                              <1>       ; must be cleared in order to prevent infinitive 'syswait'
                              <1>       ; system call loop from the application/program if it calls
                              <1>       ; 'syswait' again (mistakenly) while there is not a zombie
                              <1>       ; or running child process to wait. ('forktest.s', 17/09/2015)
                              <1>       ;
                              <1>       ; Note: syswait will return with error if there is not a
                              <1>       ;       zombie or running process to wait.
                              <1>       ;
0000CA00 6629C0               <1>       sub    ax, ax
0000CA03 668986[3E000300]     <1>       mov    [esi+p.ppid-2], ax ; 0 ; 17/09/2015
0000CA0A E9D1FCFFFF           <1>       jmp    sysret0 ; ax = 0
                              <1>       ;
                              <1>       ;jmp   sysret
                              <1>              ; br sysret1 / return cause child is dead
                              <1> syswait_2: ; 2:
0000CA0F 66D1E6               <1>       shl    si, 1
                              <1>              ; asl r2 / r2x2 to get index into p.ppid table
                              <1> syswait_3: ; 3:
0000CA12 6683FE20             <1>       cmp    si, nproc+nproc
                              <1>              ; cmp r2,$nproc+nproc / have all processes been checked?
0000CA16 72B8                 <1>       jb     short syswait_1
                              <1>              ; blt 1b / no, continue search
                              <1>       ;and   cx, cx
0000CA18 20C9                 <1>       and    cl, cl
                              <1>              ; tst r3 / one gets here if there are no children
                              <1>                    ; / or children that are still active
                              <1>       ; 30/10/2013
0000CA1A 750B                 <1>       jnz    short syswait_4
                              <1>       ;jz    error
                              <1>              ; beq error1 / there are no children, error
0000CA1C 890D[64030300]       <1>       mov    [u.r0], ecx ; 0
0000CA22 E997FCFFFF           <1>       jmp    error
                              <1> syswait_4:
0000CA27 8A1D[B3030300]       <1>       mov    bl, [u.uno]
                              <1>              ; movb u.uno,r1 / there are children so put
                              <1>                          ; / parent process number in r1
```

```
0000CA2D FE83[AF000300]      <1>       inc    byte [ebx+p.stat-1] ; 2, SWAIT, 05/02/2014
                             <1>              ; incb p.stat-1(r1) / it is waiting for
                             <1>                             ; / other children to die
                             <1>       ; 04/11/2013
0000CA33 E8671B0000          <1>       call   swap
                             <1>              ; jsr r0,swap / swap it out, because it's waiting
0000CA38 EB82                <1>       jmp    syswait_0
                             <1>              ; br syswait / wait on next process
                             <1>
                             <1> sysfork: ; < create a new process >
                             <1>       ; 02/01/2017 (TRDOS 386 modification)
                             <1>       ; 04/09/2015, 18/05/2015
                             <1>       ; 28/08/2015, 01/09/2015, 02/09/2015
                             <1>       ; 09/05/2015, 10/05/2015, 14/05/2015
                             <1>       ; 06/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 24/05/2013 - 14/02/2014 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'sysfork' creates a new process. This process is referred
                             <1>       ; to as the child process. This new process core image is
                             <1>       ; a copy of that of the caller of 'sysfork'. The only
                             <1>       ; distinction is the return location and the fact that (u.r0)
                             <1>       ; in the old process (parent) contains the process id (p.pid)
                             <1>       ; of the new process (child). This id is used by 'syswait'.
                             <1>       ; 'sysfork' works in the following manner:
                             <1>       ;    1) The process status table (p.stat) is searched to find
                             <1>       ;      a process number that is unused. If none are found
                             <1>       ;      an error occurs.
                             <1>       ;    2) when one is found, it becomes the child process number
                             <1>       ;      and it's status (p.stat) is set to active.
                             <1>       ;    3) If the parent had a control tty, the interrupt
                             <1>       ;      character in that tty buffer is cleared.
                             <1>       ;    4) The child process is put on the lowest priority run
                             <1>       ;      queue via 'putlu'.
                             <1>       ;    5) A new process name is gotten from 'mpid' (actually
                             <1>       ;      it is a unique number) and is put in the child's unique
                             <1>       ;      identifier; process id (p.pid).
                             <1>       ;    6) The process name of the parent is then obtained and
                             <1>       ;      placed in the unique identifier of the parent process
                             <1>       ;      name is then put in 'u.r0'.
                             <1>       ;    7) The child process is then written out on disk by
                             <1>       ;      'wswap',i.e., the parent process is copied onto disk
                             <1>       ;      and the child is born. (The child process is written
                             <1>       ;      out on disk/drum with 'u.uno' being the child process
                             <1>       ;      number.)
                             <1>       ;    8) The parent process number is then restored to 'u.uno'.
                             <1>       ;    9) The child process name is put in 'u.r0'.
                             <1>       ;   10) The pc on the stack sp + 18 is incremented by 2 to
                             <1>       ;      create the return address for the parent process.
                             <1>       ;   11) The 'u.fp' list as then searched to see what files
                             <1>       ;      the parent has opened. For each file the parent has
                             <1>       ;      opened, the corresponding 'fsp' entry must be updated
                             <1>       ;      to indicate that the child process also has opened
                             <1>       ;      the file. A branch to 'sysret' is then made.

                             <1>       ;
                             <1>       ; Calling sequence:
                             <1>       ;      from shell ?
                             <1>       ; Arguments:
                             <1>       ;      -
                             <1>       ; Inputs: -
                             <1>       ; Outputs: *u.r0 - child process name
                             <1>       ; ...............................................................
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 modification:
                             <1>       ;      AX = r0 = PID (>0) (at the return of 'sysfork')
                             <1>       ;      = process id of child a parent process returns
                             <1>       ;      = process id of parent when a child process returns
                             <1>       ;
                             <1>       ;       In original UNIX v1, sysfork is called and returns as
                             <1>       ;      in following manner: (with an example: c library, fork)
                             <1>       ;
                             <1>       ;      1:
                             <1>       ;            sys    fork
                             <1>       ;                   br 1f  / child process returns here
                             <1>       ;            bes    2f     / parent process returns here
                             <1>       ;            / pid of new process in r0
                             <1>       ;            rts    pc
                             <1>       ;      2: / parent process condionally branches here
                             <1>       ;            mov    $-1,r0 / pid = -1 means error return
                             <1>       ;            rts    pc
                             <1>       ;
                             <1>       ;      1: / child process brances here
                             <1>       ;            clr    r0   / pid = 0 in child process
                             <1>       ;            rts    pc
                             <1>       ;
                             <1>       ;      In UNIX v7x86 (386) by Robert Nordier (1999)
                             <1>       ;            // pid = fork();
                             <1>       ;            //
                             <1>       ;            // pid == 0 in child process;
                             <1>       ;            // pid == -1 means error return
                             <1>       ;            // in child,
                             <1>       ;            //     parents id is in par_uid if needed
                             <1>       ;
                             <1>       ;            _fork:
                             <1>       ;                   mov    $.fork,eax
                             <1>       ;                   int    $0x30
                             <1>       ;                   jmp    1f
                             <1>       ;                   jnc    2f
                             <1>       ;                   jmp    cerror
                             <1>       ;            1:
                             <1>       ;                   mov    eax,_par_uid
                             <1>       ;                   xor    eax,eax
                             <1>       ;            2:
```

```
                             <1>       ;                   ret
                             <1>       ;
                             <1>       ;     In Retro UNIX 8086 v1,
                             <1>       ;     'sysfork' returns in following manner:
                             <1>       ;
                             <1>       ;            mov    ax, sys_fork
                             <1>       ;            mov    bx, offset @f ; routine for child
                             <1>       ;            int    20h
                             <1>       ;            jc     error
                             <1>       ;
                             <1>       ;     ; Routine for parent process here (just after 'jc')
                             <1>       ;            mov    word ptr [pid_of_child], ax
                             <1>       ;            jmp    next_routine_for_parent
                             <1>       ;
                             <1>       ;     @@: ; routine for child process here
                             <1>       ;             ....
                             <1>       ;     NOTE: 'sysfork' returns to specified offset
                             <1>       ;            for child process by using BX input.
                             <1>       ;           (at first, parent process will return then
                             <1>       ;           child process will return -after swapped in-
                             <1>       ;           'syswait' is needed in parent process
                             <1>       ;           if return from child process will be waited for.)
                             <1>       ;
                             <1>
                             <1> ; / create a new process
                             <1>       ; EBX = return address for child process
                             <1>            ; (Retro UNIX 8086 v1 modification !)
0000CA3A 31F6                <1>       xor   esi, esi
                             <1>             ; clr r1
                             <1> sysfork_1: ; 1: / search p.stat table for unused process number
0000CA3C 46                  <1>       inc   esi
                             <1>             ; inc r1
0000CA3D 80BE[AF000300]00    <1>       cmp   byte [esi+p.stat-1], 0 ; SFREE, 05/02/2014
                             <1>             ; tstb p.stat-1(r1) / is process active, unused, dead
0000CA44 760B                <1>       jna   short sysfork_2
                             <1>             ; beq 1f / it's unused so branch
0000CA46 6683FE10            <1>       cmp   si, nproc
                             <1>             ; cmp r1,$nproc / all processes checked
0000CA4A 72F0                <1>       jb    short sysfork_1
                             <1>             ; blt 1b / no, branch back
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1. modification:
                             <1>       ;     Parent process returns from 'sysfork' to address
                             <1>       ;     which is just after 'sysfork' system call in parent
                             <1>       ;     process. Child process returns to address which is put
                             <1>       ;     in BX register by parent process for 'sysfork'.
                             <1>       ;
                             <1>             ;add $2,18.(sp) / add 2 to pc when trap occured, points
                             <1>                          ; / to old process return
                             <1>             ; br error1 / no room for a new process
0000CA4C E96DFCFFFF          <1>       jmp   error
                             <1> sysfork_2: ; 1:
0000CA51 E82481FFFF          <1>       call  allocate_page
0000CA56 0F8262FCFFFF        <1>       jc    error
0000CA5C 50                  <1>       push  eax   ; UPAGE (user structure page) address
                             <1>       ; Retro UNIX 386 v1 modification!
0000CA5D E82783FFFF          <1>       call  duplicate_page_dir
                             <1>             ; EAX = New page directory
0000CA62 730B                <1>       jnc   short sysfork_3
0000CA64 58                  <1>       pop   eax   ; UPAGE (user structure page) address
0000CA65 E8EE82FFFF          <1>       call  deallocate_page
0000CA6A E94FFCFFFF          <1>       jmp   error
                             <1> sysfork_3:
                             <1>       ; Retro UNIX 386 v1 modification !
0000CA6F 56                  <1>       push  esi
0000CA70 E8B81B0000          <1>       call  wswap ; save current user (u) structure, user registers
                             <1>                  ; and interrupt return components (for IRET)
0000CA75 8705[B8030300]      <1>       xchg  eax, [u.pgdir] ; page directory of the child process
0000CA7B A3[BC030300]        <1>       mov   [u.ppgdir], eax ; page directory of the parent process
0000CA80 5E                  <1>       pop   esi
0000CA81 58                  <1>       pop   eax   ; UPAGE (user structure page) address
                             <1>             ; [u.usp] = esp
0000CA82 89F7                <1>       mov   edi, esi
0000CA84 66C1E702            <1>       shl   di, 2
0000CA88 8987[BC000300]      <1>       mov   [edi+p.upage-4], eax ; memory page for 'user' struct
0000CA8E A3[B4030300]        <1>       mov   [u.upage], eax ; memory page for 'user' struct (child)
                             <1>       ; 28/08/2015
0000CA93 0FB605[B3030300]    <1>       movzx eax, byte [u.uno] ; parent process number
                             <1>             ; movb u.uno,-(sp) / save parent process number
0000CA9A 89C7                <1>       mov   edi, eax
0000CA9C 50                  <1>         push eax ; **
0000CA9D 8A87[7F000300]      <1>       mov     al, [edi+p.ttyc-1] ; console tty (parent)
                             <1>       ; 18/09/2015
                             <1>       ;mov     [esi+p.ttyc-1], al ; set child's console tty
                             <1>       ;mov     [esi+p.waitc-1], ah ; 0 ; reset child's wait channel
0000CAA3 668986[7F000300]    <1>       mov     [esi+p.ttyc-1], ax ; al - set child's console tty
                             <1>                            ; ah - reset child's wait channel
0000CAAA 89F0                <1>       mov   eax, esi
0000CAAC A2[B3030300]        <1>       mov   [u.uno], al ; child process number
                             <1>             ;movb r1,u.uno / set child process number to r1
0000CAB1 FE86[AF000300]      <1>         inc     byte [esi+p.stat-1] ; 1, SRUN, 05/02/2014
                             <1>             ; incb p.stat-1(r1) / set p.stat entry for child
                             <1>                          ; / process to active status
                             <1>             ; mov u.ttyp,r2 / put pointer to parent process'
                             <1>                         ; / control tty buffer in r2
                             <1>                 ; beq 2f / branch, if no such tty assigned
                             <1>             ; clrb 6(r2) / clear interrupt character in tty buffer
                             <1>       ; 2:
0000CAB7 53                  <1>       push  ebx  ; * return address for the child process
                             <1>                  ; * Retro UNIX 8086 v1 feature only !
                             <1>       ; (Retro UNIX 8086 v1 modification!)
                             <1>             ; mov $runq+4,r2
0000CAB8 BB[54030300]        <1>       mov   ebx, runq+2 ; normal run queue ; 02/01/2017
```

```
0000CABD E8DB1B0000          <1>       call  putlu
                             <1>             ; jsr r0,putlu / put child process on lowest priority
                             <1>                       ; / run queue
0000CAC2 66D1E6              <1>       shl   si, 1
                             <1>             ; asl r1 / multiply r1 by 2 to get index
                             <1>                    ; / into p.pid table
0000CAC5 66FF05[4E030300]    <1>       inc   word [mpid]
                             <1>             ; inc mpid / increment m.pid; get a new process name
0000CACC 66A1[4E030300]      <1>       mov   ax, [mpid]
0000CAD2 668986[1E000300]    <1>       mov   [esi+p.pid-2], ax
                             <1>             ;mov mpid,p.pid-2(r1) / put new process name
                             <1>                               ; / in child process' name slot
0000CAD9 5A                  <1>       pop   edx  ; * return address for the child process
                             <1>                  ; * Retro UNIX 8086 v1 feature only !
0000CADA 5B                  <1>       pop   ebx  ; **
                             <1>       ;mov  ebx, [esp] ; ** parent process number
                             <1>             ; movb (sp),r2 / put parent process number in r2
0000CADB 66D1E3              <1>       shl   bx, 1
                             <1>             ;asl r2 / multiply by 2 to get index into below tables
                             <1>       ;movzx eax, word [ebx+p.pid-2]
0000CADE 668B83[1E000300]    <1>       mov   ax, [ebx+p.pid-2]
                             <1>             ; mov p.pid-2(r2),r2 / get process name of parent
                             <1>                              ; / process
0000CAE5 668986[3E000300]    <1>       mov   [esi+p.ppid-2], ax
                             <1>             ; mov r2,p.ppid-2(r1) / put parent process name
                             <1>                      ; / in parent process slot for child
0000CAEC A3[64030300]        <1>       mov   [u.r0], eax
                             <1>             ; mov r2,*u.r0 / put parent process name on stack
                             <1>                         ; / at location where r0 was saved
0000CAF1 8B2D[5C030300]      <1>       mov   ebp, [u.sp] ; points to return address (EIP for IRET)
0000CAF7 895500              <1>       mov   [ebp], edx ; *, CS:EIP -> EIP
                             <1>                       ; * return address for the child process
                             <1>             ; mov $sysret1,-(sp) /
                             <1>             ; mov sp,u.usp / contents of sp at the time when
                             <1>                          ; / user is swapped out
                             <1>             ; mov $sstack,sp / point sp to swapping stack space
                             <1>       ; 04/09/2015 - 01/09/2015
                             <1>       ; [u.usp] = esp
0000CAFA 68[DEC60000]        <1>       push  sysret ; ***
0000CAFF 8925[60030300]      <1>       mov   [u.usp], esp ; points to 'sysret' address (***)
                             <1>                         ; (for child process)
0000CB05 31C0                <1>       xor   eax, eax
0000CB07 66A3[94030300]      <1>       mov   [u.ttyp], ax ; 0
                             <1>       ;
0000CB0D E81B1B0000          <1>       call  wswap ; Retro UNIX 8086 v1 modification !
                             <1>             ;jsr r0,wswap / put child process out on drum
                             <1>             ;jsr r0,unpack / unpack user stack
                             <1>             ;mov u.usp,sp / restore user stack pointer
                             <1>             ; tst (sp)+ / bump stack pointer
                             <1>       ; Retro UNIX 386 v1 modification !
0000CB12 58                  <1>       pop   eax ; ***
0000CB13 66D1E3              <1>       shl   bx, 1
0000CB16 8B83[BC000300]      <1>       mov    eax, [ebx+p.upage-4] ; UPAGE address ; 14/05/2015
0000CB1C E8441B0000          <1>       call  rswap ; restore parent process 'u' structure,
                             <1>                   ; registers and return address (for IRET)
                             <1>             ;movb (sp)+,u.uno / put parent process number in u.uno
0000CB21 0FB705[4E030300]    <1>         movzx   eax, word [mpid]
0000CB28 A3[64030300]        <1>       mov   [u.r0], eax
                             <1>             ; mov mpid,*u.r0 / put child process name on stack
                             <1>                          ; / where r0 was saved
                             <1>             ; add $2,18.(sp) / add 2 to pc on stack; gives parent
                             <1>                             ; / process return
                             <1>       ;xor  ebx, ebx
0000CB2D 31F6                <1>       xor     esi, esi
                             <1>             ;clr r1
                             <1> sysfork_4: ; 1: / search u.fp list to find the files
                             <1>            ; / opened by the parent process
                             <1>       ; 01/09/2015
                             <1>       ;xor  bh, bh
                             <1>       ;mov  bl, [esi+u.fp]
0000CB2F 8A86[6A030300]      <1>       mov   al, [esi+u.fp]
                             <1>             ; movb u.fp(r1),r2 / get an open file for this process
                             <1>         ;or      bl, bl
0000CB35 08C0                <1>       or    al, al
0000CB37 740D                <1>       jz    short sysfork_5
                             <1>             ; beq 2f / file has not been opened by parent,
                             <1>                    ; / so branch
0000CB39 B40A                <1>       mov   ah, 10 ; Retro UNIX 386 v1 fsp structure size = 10 bytes
0000CB3B F6E4                <1>       mul   ah
                             <1>       ;movzx ebx, ax
0000CB3D 6689C3              <1>       mov   bx, ax
                             <1>       ;shl     bx, 3
                             <1>             ; asl r2 / multiply by 8
                             <1>                   ; asl r2 / to get index into fsp table
                             <1>                   ; asl r2
0000CB40 FE83[4E010300]      <1>       inc    byte [ebx+fsp-2]
                             <1>             ; incb fsp-2(r2) / increment number of processes
                             <1>                         ; / using file, because child will now be
                             <1>                         ; / using this file
                             <1> sysfork_5: ; 2:
0000CB46 46                  <1>         inc     esi
                             <1>             ; inc r1 / get next open file
0000CB47 6683FE0A            <1>         cmp     si, 10
                             <1>             ; cmp r1,$10. / 10. files is the maximum number which
                             <1>                      ; / can be opened
0000CB4B 72E2                <1>       jb    short sysfork_4
                             <1>             ; blt 1b / check next entry
0000CB4D E98CFBFFFF          <1>       jmp   sysret
                             <1>             ; br sysret1
                             <1>
                             <1> syscreat: ; < create file >
                             <1>       ; 13/11/2017
                             <1>       ; 27/10/2016
                             <1>       ; 25/10/2016, 26/10/2016
                             <1>       ; 15/10/2016, 16/10/2016, 17/10/2016
                             <1>       ; 10/10/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;           -derived from INT_21H.ASM-
                             <1>       ;            ("loc_INT21h_create_file")
                             <1>         ;    10/07/2011 (12/03/2011)
                             <1>         ;    INT 21h Function AH = 3Ch
                             <1>         ;    Create File
                             <1>         ;    INPUT
                             <1>         ;       CX = Attributes
                             <1>         ;           DS:DX= Address of zero terminaned path name
                             <1>         ;
                             <1>       ; 27/12/2015 (Retro UNIX 386 v1.1)
                             <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 27/05/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'syscreat' called with two arguments; name and mode.
                             <1>       ; u.namep points to name of the file and mode is put
                             <1>       ; on the stack. 'namei' is called to get i-number of the file.
                             <1>       ; If the file aready exists, it's mode and owner remain
                             <1>       ; unchanged, but it is truncated to zero length. If the file
                             <1>       ; did not exist, an i-node is created with the new mode via
                             <1>       ; 'maknod' whether or not the file already existed, it is
                             <1>       ; open for writing. The fsp table is then searched for a free
                             <1>       ; entry. When a free entry is found, proper data is placed
                             <1>       ; in it and the number of this entry is put in the u.fp list.
                             <1>       ; The index to the u.fp (also know as the file descriptor)
                             <1>       ; is put in the user's r0.
                             <1>       ;
                             <1>       ; Calling sequence:
                             <1>       ;      syscreate; name; mode
                             <1>       ; Arguments:
                             <1>       ;      name - name of the file to be created
                             <1>       ;      mode - mode of the file to be created
                             <1>       ; Inputs: (arguments)
                             <1>       ; Outputs: *u.r0 - index to u.fp list
                             <1>       ;               (the file descriptor of new file)
                             <1>       ; ...............................................................
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 modification:
                             <1>       ;       'syscreate' system call has two arguments; so,
                             <1>       ;      * 1st argument, name is pointed to by BX register
                             <1>       ;      * 2nd argument, mode is in CX register
                             <1>       ;
                             <1>       ;      AX register (will be restored via 'u.r0') will return
                             <1>       ;      to the user with the file descriptor/number
                             <1>       ;      (index to u.fp list).
                             <1>       ;
                             <1>       ;call  arg2
                             <1>       ; * name - 'u.namep' points to address of file/path name
                             <1>       ;          in the user's program segment ('u.segmnt')
                             <1>       ;          with offset in BX register (as sysopen argument 1).
                             <1>       ; * mode - sysopen argument 2 is in CX register
                             <1>       ;          which is on top of stack.
                             <1>       ;
                             <1>       ; TRDOS 386 (10/10/2016)
                             <1>       ;
                             <1>         ; INPUT ->
                             <1>         ;       CL = File Attributes
                             <1>       ;           bit 0 (1) - Read only file (R)
                             <1>       ;             bit 1 (1) - Hidden file (H)
                             <1>         ;             bit 2 (1) - System file (R)
                             <1>       ;             bit 3 (1) - Volume label/name (V)
                             <1>         ;             bit 4 (1) - Subdirectory (D)
                             <1>       ;           bit 5 (1) - File has been archived (A)
                             <1>         ;          EBX = Pointer to filename (ASCIIZ) -path-
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;          eax = File/Device Handle/Number (index) (AL)
                             <1>       ;          cf = 1 -> Error code in AL
                             <1>       ;
                             <1>       ; Modified Registers: EAX (at the return of system call)
                             <1>       ;
                             <1>       ; Note: If the file is existing and it has not any one
                             <1>       ;      of S,H,R,V,D attributes, it will be truncated
                             <1>       ;      to zero length; otherwise, access error will be
                             <1>       ;      returned.
                             <1>
                             <1> sysmkdir_0:
0000CB52 F6C108              <1>       test   cl, 08h ; Volume name
0000CB55 740A                <1>       jz     short syscreat_0
                             <1>
                             <1>       ; Volume name or long name creation
                             <1>       ; is not permitted (in TRDOS 386)!
0000CB57 B80B000000          <1>       mov    eax, ERR_FILE_ACCESS  ; 11 ; 'permission denied !'
0000CB5C E926020000          <1>         jmp  sysopen_dev_err
                             <1>
                             <1> syscreat_0:
                             <1>         ;mov [u.namep], ebx
0000CB61 51                  <1>       push   ecx
0000CB62 89DE                <1>       mov    esi, ebx
                             <1>       ; file name is forced, change directory as temporary
                             <1>       ;mov   ax, 1
                             <1>       ;mov   [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
                             <1>       ;call  set_working_path
0000CB64 E892300000          <1>       call   set_working_path_x ; 17/10/2016
0000CB69 0F82D7000000        <1>       jc     syscreat_err
                             <1>
                             <1>       ; 16/10/2016
0000CB6F 803D[F3650100]00    <1>       cmp    byte [SWP_inv_fname], 0
0000CB76 776C                <1>       ja     short  syscreat_inv_fname ; invalid file name !
                             <1>
                             <1>       ; Here, we have a valid path and also a valid file name
                             <1>       ; (Working dir has been changed if the path
                             <1>       ;  -file name string- had contained a dir name.)
                             <1>
0000CB78 6631C0              <1>       xor    ax, ax
                             <1>       ;mov   esi, FindFile_Name
0000CB7B E8E3B6FFFF          <1>       call   find_first_file
0000CB80 59                  <1>       pop    ecx
                             <1>              ; ESI = Directory Entry (FindFile_DirEntry) Location
                             <1>              ; EDI = Directory Buffer Directory Entry Location
                             <1>              ; EAX = File Size
                             <1>              ;  BL = Attributes of The File/Directory
                             <1>              ;  BH = Long Name Yes/No Status (>0 is YES)
                             <1>              ;  DX > 0 : Ambiguous filename chars are used
0000CB81 7269                <1>       jc     short syscreat_1 ; file not found (the good!)
                             <1>                            ; or another error (the bad')
                             <1>
                             <1>       ; (& the uggly!) truncate file to zero length before open
                             <1>
                             <1>       ;'*' and '?' already checked at 'set_working_path' stage
                             <1>       ;and   dx, dx
                             <1>       ;jnz   short sysmkdir_err ; permission denied
                             <1>                              ; invalid filename chars
                             <1>
                             <1>       ;test  cl, 10h ; subdirectory ?
                             <1>       ;jnz   short sysmkdir_err
                             <1>
                             <1>       ; BL = File Attributes:
                             <1>       ;            bit 0 (1) - Read only file (R)
                             <1>       ;              bit 1 (1) - Hidden file (H)
                             <1>         ;             bit 2 (1) - System file (R)
                             <1>       ;              bit 3 (1) - Volume label/name (V)
                             <1>         ;             bit 4 (1) - Subdirectory (D)
                             <1>       ;            bit 5 (1) - File has been archived
                             <1>
                             <1>       ; * existing directory must not be truncated
                             <1>       ;   (we don't know it is empty or not, at this stage)
                             <1>       ; * existing volume name (or a long name) can not be
                             <1>       ;   re-created or truncated by 'syscreat'
                             <1>       ; * A file with S, H, R attributes must not be truncated
                             <1>       ;   (change attributes to normal, if you need truncate it)
                             <1>
0000CB83 F6C31F              <1>       test   bl, 00011111b  ; check attributes of existing file
0000CB86 754E                <1>       jnz    short sysmkdir_err
                             <1>
                             <1>       ;; normal file, OK to continue...
                             <1>
                             <1>       ; ESI = FindFile_DirEntry
0000CB88 668B4614            <1>       mov    ax, [esi+DirEntry_FstClusHI] ; 20
0000CB8C C1E010              <1>       shl    eax, 16 ; 13/11/2017
0000CB8F 668B461A            <1>       mov    ax, [esi+DirEntry_FstClusLO] ; 26
                             <1>       ; EAX = First cluster to be truncated/unlinked
0000CB93 57                  <1>       push   edi
0000CB94 51                  <1>       push   ecx
0000CB95 BE00010900          <1>       mov    esi, Logical_DOSDisks
0000CB9A 29C9                <1>       sub    ecx, ecx
0000CB9C 8A2D[FE580100]      <1>       mov    ch, [Current_Drv]
0000CBA2 01CE                <1>       add    esi, ecx
                             <1>       ; ESI = Logical dos drive description table address
0000CBA4 E8C9F7FFFF          <1>       call   truncate_cluster_chain
0000CBA9 59                  <1>       pop    ecx
0000CBAA 5F                  <1>       pop    edi
0000CBAB 7230                <1>       jc     short syscreate_truncate_err
                             <1>
                             <1>       ; 26/10/2016
                             <1>       ; EDI = Directory entry address in directory buffer
                             <1>       ; Update directory entry
0000CBAD E848DCFFFF          <1>       call   convert_current_date_time
                             <1>       ; OUTPUT -> DX = Date in dos dir entry format
                             <1>         ;        AX = Time in dos dir entry format
0000CBB2 66894716            <1>       mov    [edi+DirEntry_WrtTime], ax
0000CBB6 66895718            <1>       mov    [edi+DirEntry_WrtDate], dx
0000CBBA 66895712            <1>       mov    [edi+DirEntry_LastAccDate], dx
0000CBBE 31C0                <1>       xor    eax, eax ; file size = 0
0000CBC0 89471C              <1>       mov    [edi+DirEntry_FileSize], eax ; 0
0000CBC3 C605[28610100]02    <1>       mov    byte [DirBuff_ValidData], 2 ; data changed sign
0000CBCA BE[F4620100]        <1>       mov    esi, FindFile_DirEntry
0000CBCF B201                <1>       mov    dl, 1 ; open file for writing
0000CBD1 E9AA000000          <1>       jmp    sysopen_2
                             <1>
                             <1> sysmkdir_err:
                             <1>       ; 1 = write, 2 = read & write, >2 = invalid
0000CBD6 B80B000000          <1>         mov eax, ERR_FILE_ACCESS  ; 11 ; 'permission denied !'
0000CBDB EB73                <1>         jmp short sysopen_err
                             <1>
                             <1> syscreate_truncate_err:
0000CBDD B812000000          <1>       mov    eax, ERR_DRV_WRITE ; 18 ; 'disk write error !'
0000CBE2 EB6C                <1>         jmp short sysopen_err
                             <1>
                             <1> syscreat_inv_fname:  ; invalid file name chars
                             <1>       ; 16/10/2016
0000CBE4 B81A000000          <1>       mov    eax, ERR_INV_FILE_NAME  ; 26 ; invalid file name chars
0000CBE9 59                  <1>       pop    ecx
0000CBEA EB64                <1>       jmp    sysopen_err
                             <1>
                             <1> syscreat_1:
                             <1>       ; Error code in EAX
0000CBEC 3C02                <1>         cmp al, 02h ; 'File not found' error
0000CBEE 7560                <1>         jne sysopen_err
                             <1>
0000CBF0 F6C110              <1>       test   cl, 10h ; Directory
0000CBF3 0F852C020000        <1>       jnz    sysmkdir_2
                             <1>
                             <1> syscreat_2:
```

```
0000CBF9 BE[E4620100]        <1>       mov    esi, FindFile_Name
                             <1>         ;xor edx, edx
0000CBFE 31C0                <1>         xor  eax, eax ; File Size  = 0
0000CC00 31DB                <1>       xor    ebx, ebx
0000CC02 4B                  <1>       dec    ebx ; FFFFFFFFh -> create empty file
                             <1>                 ;              (only for FAT fs)
                             <1>       ; CL = File Attributes
0000CC03 E8F8EBFFFF          <1>       call   create_file
0000CC08 7246                <1>       jc     sysopen_err
                             <1>              ; EAX = New file's first cluster
                             <1>              ; ESI = Logical Dos Drv Descr. Table Addr.
                             <1>              ; EBX = offset CreateFile_Size
                             <1>              ; ECX = Sectors per cluster (<256)
                             <1>              ; EDX = Directory entry index/number (<65536)
                             <1>       ; 26/10/2016
                             <1>       ;mov   esi, Directory_Buffer
                             <1>       ;shl   dx, 5 ; *32
                             <1>       ;add   esi, edx
                             <1>       ;; esi = directory entry address in directory buffer
                             <1>
                             <1>       ; Here, directory entry has been created but last
                             <1>       ; modification date & time of the parent dir has not
                             <1>       ; been updated, yet!
                             <1>       ; (Note: Directory and FAT buffers have been updated...)
                             <1>
0000CC0A E824DDFFFF          <1>       call   update_parent_dir_lmdt ; now, it is OK too!
                             <1>
                             <1>       ; 25/10/2016
0000CC0F 66B80018            <1>       mov    ax, 1800h
0000CC13 BE[E4620100]        <1>       mov    esi, FindFile_Name
0000CC18 E846B6FFFF          <1>       call   find_first_file
0000CC1D 7231                <1>       jc     short sysopen_err
                             <1>
                             <1>       ; Only possible error after here is
                             <1>       ; "too many open files !" error.
                             <1>       ;
                             <1>       ; If "syscreat" will return with that error,
                             <1>       ; (the file has been created but it could not be opened)
                             <1>       ; the user must retry to open this file again
                             <1>       ; or must close another file before using
                             <1>       ; "sysopen" system call.
                             <1>
0000CC1F B201                <1>       mov    dl, 1 ; open file for writing
                             <1>       ; ESI = Directory Entry (FindFile_DirEntry) Location
                             <1>       ; EAX = File Size (= 0)
0000CC21 EB5D                <1>       jmp    short sysopen_2
                             <1>
                             <1> sysopen: ;<open file>
                             <1>       ; 26/10/2016
                             <1>       ; 24/10/2016
                             <1>       ; 17/10/2016
                             <1>       ; 15/10/2016
                             <1>       ; 06/10/2016, 07/10/2016, 08/10/2016
                             <1>       ; 05/10/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;          -derived from INT_21H.ASM-
                             <1>       ;            ("loc_INT21h_open_file")
                             <1>         ;    26/02/2011
                             <1>         ;    INT 21h Function AH = 3Dh
                             <1>         ;    Open File
                             <1>         ;    INPUT
                             <1>         ;       AL= File Access Value
                             <1>       ;          0- Open for reading
                             <1>       ;            1- Open for writing
                             <1>         ;            2- Open for reading and writing
                             <1>         ;          DS:DX= Pointer to filename (ASCIIZ)
                             <1>         ;
                             <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 22/05/2013 - 27/05/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'sysopen' opens a file in following manner:
                             <1>       ;    1) The second argument in a sysopen says whether to
                             <1>       ;      open the file ro read (0) or write (>0).
                             <1>       ;    2) I-node of the particular file is obtained via 'namei'.
                             <1>       ;    3) The file is opened by 'iopen'.
                             <1>       ;    4) Next housekeeping is performed on the fsp table
                             <1>       ;      and the user's open file list - u.fp.
                             <1>       ;      a) u.fp and fsp are scanned for the next available slot.
                             <1>       ;      b) An entry for the file is created in the fsp table.
                             <1>       ;      c) The number of this entry is put on u.fp list.
                             <1>       ;      d) The file descriptor index to u.fp list is pointed
                             <1>       ;         to by u.r0.
                             <1>       ;
                             <1>       ; Calling sequence:
                             <1>       ;      sysopen; name; mode
                             <1>       ; Arguments:
                             <1>       ;      name - file name or path name
                             <1>       ;      mode - 0 to open for reading
                             <1>       ;             1 to open for writing
                             <1>       ; Inputs: (arguments)
                             <1>       ; Outputs: *u.r0 - index to u.fp list (the file descriptor)
                             <1>       ;               is put into r0's location on the stack.
                             <1>       ; ..............................................................
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 modification:
                             <1>       ;       'sysopen' system call has two arguments; so,
                             <1>       ;      * 1st argument, name is pointed to by BX register
                             <1>       ;      * 2nd argument, mode is in CX register
                             <1>       ;
                             <1>       ;      AX register (will be restored via 'u.r0') will return
                             <1>       ;      to the user with the file descriptor/number
                             <1>       ;      (index to u.fp list).
                             <1>       ;
                             <1>       ;call  arg2
```

```
                            <1>         ; * name - 'u.namep' points to address of file/path name
                            <1>         ;          in the user's program segment ('u.segmnt')
                            <1>         ;          with offset in BX register (as sysopen argument 1).
                            <1>         ; * mode - sysopen argument 2 is in CX register
                            <1>         ;          which is on top of stack.
                            <1>         ;
                            <1>         ; jsr r0,arg2 / get sys args into u.namep and on stack
                            <1>         ;
                            <1>                ; system call registers: ebx, ecx (through 'sysenter')
                            <1>         ;
                            <1>         ; TRDOS 386 (05/10/2016)
                            <1>         ;
                            <1>           ; INPUT ->
                            <1>           ;       CL = File Access Value (Open Mode)
                            <1>         ;            0 - Open file for reading
                            <1>         ;              1 - Open file for writing
                            <1>           ;              2 - Open device for reading
                            <1>         ;            3 - Open device for writing
                            <1>           ;        EBX = Pointer to filename/devicename (ASCIIZ)
                            <1>         ; OUTPUT ->
                            <1>         ;          eax = File/Device Handle/Number (index) (AL)
                            <1>         ;          cf = 1 -> Error code in AL
                            <1>         ;
                            <1>         ; Modified Registers: EAX (at the return of system call)
                            <1>         ;
                            <1>
0000CC23 80F901             <1>         cmp    cl, 1 ; read file (0), write file (1)
0000CC26 7614               <1>         jna    short sysopen_0
                            <1>
0000CC28 80F903             <1>         cmp    cl, 3
0000CC2B 0F8640010000       <1>         jna    sysopen_device
                            <1>
                            <1>         ; Invalid access code
0000CC31 B817000000         <1>         mov    eax, ERR_INV_PARAMETER
0000CC36 0F874B010000       <1>         ja     sysopen_dev_err
                            <1>
                            <1> sysopen_0:
                            <1>         ;mov   [u.namep], ebx
0000CC3C 51                 <1>         push   ecx
0000CC3D 89DE               <1>         mov    esi, ebx
                            <1>         ; file name is forced, change directory as temporary
                            <1>         ;mov   ax, 1
                            <1>         ;mov   [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
                            <1>         ;call  set_working_path
0000CC3F E8B72F0000         <1>         call   set_working_path_x ; 17/10/2016
0000CC44 731E               <1>         jnc    short sysopen_1
                            <1>
                            <1> syscreat_err: ; ecx = file attributes (for 'syscreat')
0000CC46 59                 <1>         pop    ecx ; open mode
0000CC47 21C0               <1>         and    eax, eax  ; 0 -> Bad Path!
0000CC49 7505               <1>         jnz    short sysopen_err
                            <1>         ; eax = 0
0000CC4B B80C000000         <1>         mov    eax, ERR_DIR_NOT_FOUND ; Directory not found !
                            <1> sysopen_err:
0000CC50 A3[64030300]       <1>         mov    [u.r0], eax
0000CC55 A3[C8030300]       <1>         mov    [u.error], eax
0000CC5A E871300000         <1>         call   reset_working_path
0000CC5F E95AFAFFFF         <1>         jmp    error
                            <1>
                            <1> sysopen_1:
                            <1>         ;mov   esi, FindFile_Name
0000CC64 66B80018           <1>           mov  ax, 1800h ; Only files
0000CC68 E8F6B5FFFF         <1>         call   find_first_file
0000CC6D 5A                 <1>         pop    edx
0000CC6E 72E0               <1>         jc     short sysopen_err ; eax = 2 (File not found !)
                            <1>
                            <1>         ; check_open_file_attr_access_code
                            <1>
0000CC70 F6C307             <1>           test bl, 7  ; system, hidden, readonly
0000CC73 740B               <1>           jz   short sysopen_2
                            <1>
0000CC75 20D2               <1>         and    dl, dl ; 0 = read mode
0000CC77 7407               <1>         jz     short sysopen_2
                            <1>
                            <1>         ; 1 = write, 2 = read & write, >2 = invalid
0000CC79 B80B000000         <1>           mov eax, ERR_FILE_ACCESS  ; 11 = 'permission denied !'
0000CC7E EBD0               <1>           jmp short sysopen_err
                            <1>
                            <1> sysopen_2:
                            <1>         ; esi = Directory Entry (FindFile_DirEntry) Location
0000CC80 89F3               <1>         mov    ebx, esi
0000CC82 31F6               <1>           xor     esi, esi ; 0
0000CC84 31FF               <1>           xor     edi, edi ; 0
                            <1> sysopen_3: ; scan the list of entries in fsp table
0000CC86 80BE[6A030300]00   <1>           cmp     byte [esi+u.fp], 0
0000CC8D 760F               <1>           jna     short sysopen_4 ; empty slot
0000CC8F 6646               <1>           inc     si
0000CC91 6683FE0A           <1>           cmp     si, 10
0000CC95 72EF               <1>         jb     short sysopen_3
                            <1> toomanyf:
0000CC97 B80D000000         <1>         mov    eax, ERR_TOO_MANY_FILES ; too many open files !
0000CC9C EBB2               <1>         jmp    short sysopen_err
                            <1>
                            <1> sysopen_4:
0000CC9E 80BF[62690100]00   <1>           cmp     byte [edi+OF_MODE], 0 ; Scan open files table
0000CCA5 760A               <1>         jna     short sysopen_5
0000CCA7 6647               <1>         inc    di
0000CCA9 6683FF0A           <1>         cmp     di, OPENFILES ; max. number of open files (=10)
0000CCAD 72EF               <1>         jb     short sysopen_4
0000CCAF EBE6               <1>         jmp    short toomanyf
                            <1>
                            <1> sysopen_5:
0000CCB1 FEC2               <1>         inc    dl
```

```
0000CCB3 8897[62690100]      <1>         mov     [edi+OF_MODE], dl
0000CCB9 8A15[A2620100]      <1>       mov    dl, [FindFile_Drv]
0000CCBF 8897[58690100]      <1>         mov     [edi+OF_DRIVE], dl ; Logical DOS drive number
0000CCC5 66C1E702            <1>       shl    di, 2 ; *4 (dword offset)
                             <1> 
0000CCC9 8987[A8690100]      <1>       mov    [edi+OF_SIZE], eax ; File size in bytes
                             <1> 
0000CCCF 668B4314            <1>         mov ax, [ebx+DirEntry_FstClusHI]
0000CCD3 C1E010              <1>       shl    eax, 16
0000CCD6 668B431A            <1>       mov    ax, [ebx+DirEntry_FstClusLO]
0000CCDA 8987[30690100]      <1>       mov     [edi+OF_FCLUSTER], eax ; First cluster
0000CCE0 8987[486A0100]      <1>       mov     [edi+OF_CCLUSTER], eax ; Current cluster
                             <1> 
0000CCE6 31DB                <1>         xor ebx, ebx
0000CCE8 899F[80690100]      <1>         mov     [edi+OF_POINTER], ebx ; offset pointer (0)
0000CCEE 899F[706A0100]      <1>         mov     [edi+OF_CCINDEX], ebx ; cluster index (0)
                             <1> 
0000CCF4 A1[14630100]        <1>       mov    eax, [FindFile_DirFirstCluster]
0000CCF9 8987[D0690100]      <1>       mov    [edi+OF_DIRFCLUSTER], eax
                             <1> 
0000CCFF A1[18630100]        <1>       mov    eax, [FindFile_DirCluster]
0000CD04 8987[F8690100]      <1>       mov    [edi+OF_DIRCLUSTER], eax
                             <1> 
                             <1>       ; Get (& Save) Volume ID
                             <1>       ; Important for files of removable drives
                             <1>       ; (In order to check the drive has same volume/disk)
0000CD0A 88D7                <1>       mov    bh, dl
0000CD0C 81C300010900        <1>         add ebx, Logical_DOSDisks
0000CD12 8A4303              <1>         mov al, [ebx+LD_FATType]
0000CD15 3C01                <1>         cmp al, 1
0000CD17 7209                <1>         jb  short sysopen_6_fs
0000CD19 3C02                <1>         cmp al, 2
0000CD1B 770A                <1>         ja  short sysopen_6_fat32
                             <1> sysopen_6_fat:
0000CD1D 8B432D              <1>         mov eax, [ebx+LD_BPB+VolumeID]
0000CD20 EB08                <1>         jmp short sysopen_7
                             <1> sysopen_6_fs:
0000CD22 8B4328              <1>         mov eax, [ebx+LD_FS_VolumeSerial]
0000CD25 EB03                <1>         jmp short sysopen_7
                             <1> sysopen_6_fat32:
0000CD27 8B4349              <1>         mov eax, [ebx+LD_BPB+FAT32_VolID]
                             <1> sysopen_7:
0000CD2A A3[F4580100]        <1>         mov [Current_VolSerial], eax
                             <1> 
0000CD2F 8987[206A0100]      <1>       mov    [edi+OF_VOLUMEID], eax
                             <1> 
                             <1>       ; 24/10/2016
0000CD35 66D1EF              <1>       shr    di, 1 ; 4/2, word offset
0000CD38 668B1D[1C630100]    <1>       mov    bx, [FindFile_DirEntryNumber]
0000CD3F 66899F[986A0100]    <1>       mov    [edi+OF_DIRENTRY], bx
                             <1> 
0000CD46 31D2                <1>       xor    edx, edx
                             <1>       ;shr   di, 2 ; /4 (byte offset)
0000CD48 66D1EF              <1>       shr    di, 1 ; 2/2, byte offset
0000CD4B 8897[76690100]      <1>       mov    byte [edi+OF_OPENCOUNT], dl ; 0
0000CD51 8897[6C690100]      <1>       mov    byte [edi+OF_STATUS], dl ; 0
                             <1> 
0000CD57 89FB                <1>       mov    ebx, edi
0000CD59 FEC3                <1>       inc    bl
                             <1> 
0000CD5B 889E[6A030300]      <1>         mov     [esi+u.fp], bl ; Open File Entry Number
0000CD61 8935[64030300]      <1>       mov     [u.r0], esi ; move index to u.fp list
                             <1>                        ; into eax on stack
                             <1> 
0000CD67 E8642F0000          <1>         call       reset_working_path
                             <1> 
0000CD6C E96DF9FFFF          <1>       jmp    sysret
                             <1> 
                             <1>       ; (Retro UNIX 386 v1.0)
                             <1>       ; 'fsp' table (10 bytes/entry)
                             <1>       ; bit 15                            bit 0
                             <1>       ; ---|-------------------------------------------
                             <1>       ; r/w|       i-number of open file
                             <1>       ; ---|-------------------------------------------
                             <1>       ;               device number
                             <1>       ; -----------------------------------------------
                             <1>       ; offset pointer, r/w pointer to file (bit 0-15)
                             <1>       ; -----------------------------------------------
                             <1>       ; offset pointer, r/w pointer to file (bit 16-31)
                             <1>       ; ----------------------|------------------------
                             <1>       ;  flag that says file    | number of processes
                             <1>       ;   has been deleted      | that have file open
                             <1>       ; ----------------------|------------------------
                             <1> 
                             <1> sysopen_device:
                             <1>       ; 15/10/2016
                             <1>       ; 08/10/2016
                             <1>       ; 07/10/2016 (TRDOS 386 = TRDOS v2.0)
0000CD71 51                  <1>       push   ecx ; open mode
0000CD72 89E5                <1>       mov    ebp, esp
0000CD74 B910000000          <1>       mov    ecx, 16 ; transfer length = 16 bytes
0000CD79 29CC                <1>       sub    esp, ecx
0000CD7B 89E7                <1>       mov    edi, esp ; destination address
0000CD7D 89DE                <1>       mov    esi, ebx ; dev name in user's memory space
0000CD7F E83F1A0000          <1>       call   transfer_from_user_buffer
0000CD84 7310                <1>       jnc    short sysopen_dev_0
                             <1>       ; eax = ERR_OUT_OF_MEMORY = 4 = ERR_MINOR_IM
0000CD86 59                  <1>       pop    ecx
                             <1> sysopen_dev_err:
0000CD87 A3[64030300]        <1>       mov    [u.r0], eax
0000CD8C A3[C8030300]        <1>       mov    [u.error], eax
0000CD91 E928F9FFFF          <1>       jmp    error
                             <1> sysopen_dev_0:
```

```
0000CD96 89FE                <1>       mov    esi, edi ; Device name addr (max. 16 bytes, ASCIIZ)
                             <1>                     ; for example: "tty, TTY, /dev/tty"
0000CD98 E8DB310000          <1>       call   get_device_number
0000CD9D 89EC                <1>       mov    esp, ebp
0000CD9F 59                  <1>       pop    ecx
0000CDA0 7307                <1>       jnc    short sysopen_dev_1
0000CDA2 B818000000          <1>       mov    eax, ERR_INV_DEV_NAME ; 24 ; 'invalid device name !'
0000CDA7 EBDE                <1>       jmp    short sysopen_dev_err
                             <1> sysopen_dev_1:
                             <1>       ; eax = Device Number (AL)
                             <1>       ;  cl = Open mode (2 = device read, 3 = device write)
0000CDA9 31DB                <1>         xor     ebx, ebx ; 0
                             <1> sysopen_dev_2: ; scan the list of entries
0000CDAB 389B[6A030300]      <1>         cmp     [ebx+u.fp], bl ; 0
0000CDB1 760E                <1>         jna     short sysopen_dev_3 ; empty slot
0000CDB3 FEC3                <1>         inc     bl
0000CDB5 80FB0A              <1>         cmp     bl, 10
0000CDB8 72F1                <1>       jb     short sysopen_dev_2
                             <1>       ;
0000CDBA B80D000000          <1>       mov    eax, ERR_TOO_MANY_FILES ; too many open files !
0000CDBF EBC6                <1>       jmp    short sysopen_dev_err
                             <1> sysopen_dev_3:
0000CDC1 891D[64030300]      <1>       mov    [u.r0], ebx ; File/Device index/handle/descriptor
                             <1>       ; eax = device number (entry offset)
0000CDC7 8AA8[F4660100]      <1>       mov    ch, [eax+DEV_ACCESS] ; bit 0 = accessable by users
                             <1>                                ; bit 1 = read access perm
                             <1>                                ; bit 2 = write access perm
                             <1>                                ; bit 3 = IOCTL permit to users
                             <1>                                ; bit 4 = block device if set
                             <1>                                ; bit 5 = 16 bit or 1024 byte
                             <1>                                ; bit 6 = 32 bit or 2048 byte
                             <1>                                ; bit 7 = installable device drv
0000CDCD F6C501              <1>       test   ch, 1 ; accessable by normal users (except root)
0000CDD0 7510                <1>       jnz    short sysopen_dev_4 ; yes, permission has been given
0000CDD2 803D[B0030300]00    <1>       cmp    byte [u.uid], 0 ; root?
0000CDD9 7607                <1>       jna    short sysopen_dev_4 ; superuser can open all devices
                             <1> sysopen_dev_perm_err:
0000CDDB B80B000000          <1>       mov    eax, ERR_DEV_ACCESS  ; 11 = 'permission denied !'
0000CDE0 EBA5                <1>       jmp    short sysopen_dev_err
                             <1> sysopen_dev_4:
0000CDE2 D0ED                <1>       shr    ch, 1 ; result: 1 = read, 2 = write, 3 = r & w
0000CDE4 FEC9                <1>       dec    cl  ; result: 1 = read, 2 = write
0000CDE6 84E9                <1>       test   cl, ch
0000CDE8 74F1                <1>       jz     short sysopen_dev_perm_err
                             <1>
0000CDEA D0E5                <1>       shl    ch, 1 ; bit 0 = 0
                             <1>       ; eax = device number (entry offset)
0000CDEC E8A3320000          <1>       call   device_open
0000CDF1 72E8                <1>       jc     short sysopen_dev_perm_err
                             <1>
                             <1>       ; eax = device number (entry offset)
0000CDF3 0C80                <1>       or     al, 80h ; set device bit (set bit 7 to 1)
0000CDF5 8B1D[64030300]      <1>       mov    ebx, [u.r0]
0000CDFB 8883[6A030300]      <1>       mov    [ebx+u.fp], al      ; bit 7 (=1) points to device
                             <1>
0000CE01 E9D8F8FFFF          <1>       jmp    sysret
                             <1>
                             <1> sysmkdir: ; < make directory >
                             <1>       ; 15/10/2016
                             <1>       ; 10/10/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;          -derived from INT_21H.ASM-
                             <1>       ;            ("loc_INT21h_create_file")
                             <1>         ;    10/07/2011 (12/03/2011)
                             <1>         ;    INT 21h Function AH = 3Ch
                             <1>         ;    Create File
                             <1>         ;    INPUT
                             <1>         ;       CX = Attributes
                             <1>         ;          DS:DX= Address of zero terminaned path name
                             <1>         ;
                             <1>         ;
                             <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 27/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'sysmkdir' creates an empty directory whose name is
                             <1>       ; pointed to by arg 1. The mode of the directory is arg 2.
                             <1>       ; The special entries '.' and '..' are not present.
                             <1>       ; Errors are indicated if the directory already exists or
                             <1>       ; user is not the super user.
                             <1>       ;
                             <1>       ; Calling sequence:
                             <1>       ;      sysmkdir; name; mode
                             <1>       ; Arguments:
                             <1>       ;      name - points to the name of the directory
                             <1>       ;      mode - mode of the directory
                             <1>       ; Inputs: (arguments)
                             <1>       ; Outputs: -
                             <1>       ;    (sets 'directory' flag to 1;
                             <1>       ;    'set user id on execution' and 'executable' flags to 0)
                             <1>       ; ...............................................................
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 modification:
                             <1>       ;       'sysmkdir' system call has two arguments; so,
                             <1>       ;      * 1st argument, name is pointed to by BX register
                             <1>       ;      * 2nd argument, mode is in CX register
                             <1>       ;
                             <1>       ; TRDOS 386 (10/10/2016)
                             <1>       ;
                             <1>         ; INPUT ->
                             <1>         ;       CL = Directory Attributes
                             <1>       ;            bit 0 (1) - Read only file/dir (R)
                             <1>       ;             bit 1 (1) - Hidden file/dir (H)
                             <1>         ;             bit 2 (1) - System file/dir (R)
                             <1>       ;             bit 3 (1) - Volume label/name (V)
                             <1>         ;             bit 4 (1) - Subdirectory (D)
                             <1>       ;           bit 5 (1) - File/Dir has been archived (A)
                             <1>       ;        CX = 0 -> create normal directory
                             <1>         ;          EBX = Pointer to directory name (ASCIIZ) -path-
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;          eax = First cluster of the new directory
                             <1>       ;          cf = 1 -> Error code in AL
                             <1>       ;
                             <1>       ; Modified Registers: EAX (at the return of system call)
                             <1>       ;
                             <1>       ; Note: If the file or directory is existing
                             <1>       ;     an access error will be returned.
                             <1>
0000CE06 6621C9              <1>       and    cx, cx ; if cx = 0 -> create a normal subdir
0000CE09 7413                <1>       jz     short sysmkdir_1
                             <1>
0000CE0B F6C110              <1>       test   cl, 10h ; if dir flags set, also use other flags
0000CE0E 0F853EFDFFFF        <1>       jnz    sysmkdir_0 ; jump to head of 'syscreat'
                             <1>
                             <1>       ; CX has wrong flags
0000CE14 B817000000          <1>       mov    eax, ERR_INV_FLAGS
0000CE19 E969FFFFFF          <1>       jmp    sysopen_dev_err
                             <1>
                             <1> sysmkdir_1:
0000CE1E B110                <1>       mov    cl, 10h ; set subdir flag and reset other flags
0000CE20 E92DFDFFFF          <1>       jmp    sysmkdir_0 ; jump to head of 'syscreat'
                             <1> sysmkdir_2:
                             <1>       ; jump from 'syscreat' ; from 'syscreat_1'
                             <1>       ;  CL = Directory attributes/flags
0000CE25 BE[E4620100]        <1>       mov    esi, FindFile_Name
0000CE2A E804D7FFFF          <1>       call   make_sub_directory
0000CE2F 0F821BFEFFFF        <1>       jc     sysopen_err       ; NOTE: Old type (TRDOS 8086)
                             <1>                            ; error codes must be modified
                             <1>                            ; for next TRDOS 386 versions
                             <1>                            ; (10/10/2016)
                             <1>                            ; Old (MSDOS type)
                             <1>                            ; error codes (2011):
                             <1>                            ;  2 = file not found
                             <1>                            ;  3 = directory not found
                             <1>                            ;  5 = access denied
                             <1>                            ; 12 = no more files
                             <1>                              ; 19 = disk write protected
                             <1>                            ; 39 = insufficient disk space
                             <1>                            ; 'sysdefs.s' ; 10/10/2016
                             <1>
0000CE35 A3[64030300]        <1>       mov    [u.r0], eax ; New sub dir's first cluster
                             <1>
0000CE3A E8912E0000          <1>         call       reset_working_path
                             <1>
0000CE3F E99AF8FFFF          <1>       jmp    sysret
                             <1>
                             <1> sysclose: ;<close file>
                             <1>       ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>       ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 22/05/2013 - 26/05/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'sysclose', given a file descriptor in 'u.r0', closes the
                             <1>       ; associated file. The file descriptor (index to 'u.fp' list)
                             <1>       ; is put in r1 and 'fclose' is called.
                             <1>       ;
                             <1>       ; Calling sequence:
                             <1>       ;     sysclose
                             <1>       ; Arguments:
                             <1>       ;     -
                             <1>       ; Inputs: *u.r0 - file descriptor
                             <1>       ; Outputs: -
                             <1>       ; ..............................................................
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 modification:
                             <1>       ;      The user/application program puts file descriptor
                             <1>       ;        in BX register as 'sysclose' system call argument.
                             <1>       ;      (argument transfer method 1)
                             <1>
                             <1>       ; TRDOS 386 (06/10/2016)
                             <1>       ;
                             <1>         ; INPUT ->
                             <1>         ;      EBX = File Handle/Number (file index) (AL)
                             <1>       ; OUTPUT ->
                             <1>       ;          cf = 0 -> EAX = 0
                             <1>       ;          cf = 1 -> Error code in EAX (ERR_FILE_NOT_OPEN)
                             <1>       ;
                             <1>       ; Modified Registers: EAX (at the return of system call)
                             <1>       ;
                             <1>
0000CE44 89D8                <1>       mov    eax, ebx
0000CE46 31DB                <1>       xor    ebx, ebx
0000CE48 891D[64030300]      <1>       mov    [u.r0], ebx ; 0  ; return value of EAX
0000CE4E E8450E0000          <1>       call   fclose
0000CE53 0F8385F8FFFF        <1>       jnc    sysret
0000CE59 B80A000000          <1>       mov    eax, ERR_FILE_NOT_OPEN ; file not open !
0000CE5E A3[C8030300]        <1>       mov    [u.error], eax ;
0000CE63 A3[64030300]        <1>       mov    [u.r0], eax ; ! invalid handle !
0000CE68 E951F8FFFF          <1>       jmp    error
                             <1>
                             <1> sysread: ; < read from file >
                             <1>       ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;          -derived from INT_21H.ASM-
                             <1>       ;            ("loc_INT21h_read_file")
                             <1>         ;   13/03/2011 (05/03/2011)
                             <1>         ;   INT 21h Function AH = 3Fh
                             <1>         ;   Read from a File
                             <1>         ;   INPUT
                             <1>       ;        BX = File Handle
                             <1>         ;      CX = Number of bytes to read
                             <1>         ;          DS:DX= Buffer address
                             <1>         ;
                             <1>       ; Note: TRDOS 386 'sysread' has been derived from
                             <1>       ;      Retro UNIX 386 v1 'sysread', except a few
                             <1>       ;      code modifications.
                             <1>       ;
                             <1>       ; 13/05/2015 (Retro UNIX 386 v1)
                             <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 23/05/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'sysread' is given a buffer to read into and the number of
                             <1>       ; characters to be read. If finds the file from the file
                             <1>       ; descriptor located in *u.r0 (r0). This file descriptor
                             <1>       ; is returned from a successful open call (sysopen).
                             <1>       ; The i-number of file is obtained via 'rw1' and the data
                             <1>       ; is read into core via 'readi'.
                             <1>       ;
                             <1>       ; Calling sequence:
                             <1>       ;      sysread; buffer; nchars
                             <1>       ; Arguments:
                             <1>       ;      buffer - location of contiguous bytes where
                             <1>       ;             input will be placed.
                             <1>       ;      nchars - number of bytes or characters to be read.
                             <1>       ; Inputs: *u.r0 - file descriptor (& arguments)
                             <1>       ; Outputs: *u.r0 - number of bytes read.
                             <1>       ; ...............................................................
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 modification:
                             <1>       ;       'sysread' system call has three arguments; so,
                             <1>       ;      * 1st argument, file descriptor is in BX register
                             <1>       ;      * 2nd argument, buffer address/offset in CX register
                             <1>       ;      * 3rd argument, number of bytes is in DX register
                             <1>       ;
                             <1>       ;      AX register (will be restored via 'u.r0') will return
                             <1>       ;      to the user with number of bytes read.
                             <1>       ;
                             <1>       ; TRDOS 386 (05/10/2016)
                             <1>       ;
                             <1>         ; INPUT ->
                             <1>         ;      EBX = File handle (descriptor/index)
                             <1>       ;        ECX = Buffer address
                             <1>         ;          EDX = Number of bytes
                             <1>       ; OUTPUT ->
                             <1>       ;          EAX = Number of bytes have been read
                             <1>       ;          cf = 1 -> Error code in AL
                             <1>       ;
                             <1>       ; Modified Registers: EAX (at the return of system call)
                             <1>       ;
                             <1>
                             <1>       ; EBX = File descriptor
0000CE6D E8740E0000          <1>       call   getf1
0000CE72 7277                <1>       jc     short device_read ; read data from device
                             <1>       ; EAX = First cluster of the file
                             <1>
0000CE74 E83F000000          <1>       call   rw1
0000CE79 730A                <1>       jnc    short sysread_0
                             <1>
0000CE7B A3[64030300]        <1>       mov    [u.r0], eax ; error code
0000CE80 E939F8FFFF          <1>       jmp    error
                             <1>
                             <1> sysread_0:
0000CE85 E825140000          <1>       call   readi
0000CE8A EB1D                <1>       jmp    short rw0
                             <1>
                             <1> syswrite: ; < write to file >
                             <1>       ; 23/10/2016
                             <1>       ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;          -derived from INT_21H.ASM-
                             <1>       ;            ("loc_INT21h_write_file")
                             <1>         ;   13/03/2011 (05/03/2011)
                             <1>         ;   INT 21h Function AH = 40h
                             <1>         ;   Write to a File
                             <1>         ;   INPUT
                             <1>       ;        BX = File Handle
                             <1>         ;      CX = Number of bytes to write
                             <1>         ;          DS:DX= Buffer address
                             <1>         ;
                             <1>       ; Note: TRDOS 386 'sysrwrite' has been derived from
                             <1>       ;      Retro UNIX 386 v1 'syswrite', except a few
                             <1>       ;      code modifications.
                             <1>       ;
                             <1>
                             <1>       ; 13/05/2015 (Retro UNIX 386 v1)
                             <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 23/05/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'syswrite' is given a buffer to write onto an output file
                             <1>       ; and the number of characters to write. If finds the file
                             <1>       ; from the file descriptor located in *u.r0 (r0). This file
                             <1>       ; descriptor is returned from a successful open or create call
                             <1>       ; (sysopen or syscreat). The i-number of file is obtained via
                             <1>       ; 'rw1' and buffer is written on the output file via 'write'.
                             <1>       ;
                             <1>       ; Calling sequence:
                             <1>       ;      syswrite; buffer; nchars
                             <1>       ; Arguments:
                             <1>       ;      buffer - location of contiguous bytes to be writtten.
                             <1>       ;      nchars - number of characters to be written.
                             <1>       ; Inputs: *u.r0 - file descriptor (& arguments)
                             <1>       ; Outputs: *u.r0 - number of bytes written.
```

```
                              <1>       ; ..........................................................
                              <1>       ;
                              <1>       ; Retro UNIX 8086 v1 modification:
                              <1>       ;       'syswrite' system call has three arguments; so,
                              <1>       ;      * 1st argument, file descriptor is in BX register
                              <1>       ;      * 2nd argument, buffer address/offset in CX register
                              <1>       ;      * 3rd argument, number of bytes is in DX register
                              <1>       ;
                              <1>       ;      AX register (will be restored via 'u.r0') will return
                              <1>       ;      to the user with number of bytes written.
                              <1>       ;
                              <1>       ; INPUT ->
                              <1>         ;       EBX = File handle (descriptor/index)
                              <1>       ;         ECX = Buffer address
                              <1>         ;          EDX = Number of bytes
                              <1>       ; OUTPUT ->
                              <1>       ;          EAX = Number of bytes have been written
                              <1>       ;          cf = 1 -> Error code in AL
                              <1>       ;
                              <1>       ; Modified Registers: EAX (at the return of system call)
                              <1>       ;
                              <1>
                              <1>       ; EBX = File descriptor
0000CE8C E8550E0000           <1>       call   getf1
0000CE91 7274                 <1>       jc     short device_write ; write data to device
                              <1>       ; EAX = First cluster of the file
                              <1>       ; EBX = File number  (Open file number) ; 23/10/2016
                              <1>
0000CE93 E820000000           <1>       call   rw1
0000CE98 730A                 <1>       jnc    short syswrite_0
0000CE9A A3[64030300]         <1>       mov    [u.r0], eax ; error code
0000CE9F E91AF8FFFF           <1>       jmp    error
                              <1>
                              <1> syswrite_0:
0000CEA4 E8321B0000           <1>       call   writei
                              <1> rw0: ; 1:
0000CEA9 A1[8C030300]         <1>         mov eax, [u.nread]
0000CEAE A3[64030300]         <1>       mov    [u.r0], eax
0000CEB3 E926F8FFFF           <1>       jmp    sysret
                              <1>
                              <1> rw1:
                              <1>       ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
                              <1>       ; 14/05/2015 (Retro UNIX 386 v1)
                              <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>       ; 23/05/2013 - 24/05/2013 (Retro UNIX 8086 v1)
                              <1>       ; System call registers: ebx, ecx, edx (through 'sysenter')
                              <1>       ;
                              <1>       ; EBX = File descriptor
                              <1>       ;call  getf1 ; calling point in 'getf' from 'rw1'
                              <1>       ;jc    short device_rw ; read/write data from/to device
                              <1>       ; EAX = First cluster of the file
                              <1>
0000CEB8 83F802               <1>       cmp    eax, 2
0000CEBB 7217                 <1>       jb     short rw2
                              <1>       ;
0000CEBD 890D[84030300]       <1>       mov    [u.base], ecx       ; buffer address/offset
                              <1>                           ;(in the user's virtual memory space)
0000CEC3 8915[88030300]       <1>       mov    [u.count], edx
                              <1>
0000CEC9 C705[C8030300]0000-  <1>         mov     dword [u.error], 0 ; reset the last error code
0000CED1 0000                 <1>
0000CED3 C3                   <1>       retn
                              <1>
                              <1> rw2:
0000CED4 B80A000000           <1>       mov    eax, ERR_FILE_NOT_OPEN ; file not open !
0000CED9 A3[C8030300]         <1>       mov    dword [u.error], eax
0000CEDE C3                   <1>       retn
                              <1> rw3:
0000CEDF B80B000000           <1>       mov    eax, ERR_FILE_ACCESS ; permission denied !
0000CEE4 A3[C8030300]         <1>       mov    dword [u.error], eax
0000CEE9 F9                   <1>       stc
0000CEEA C3                   <1>       retn
                              <1>
                              <1> device_read:
                              <1>       ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
                              <1>       ; cl = DEV_OPENMODE ; open mode
                              <1>       ; ch = DEV_ACCESS   ; access flags
                              <1>       ; al = DEV_DRIVER   ; device number (eax)
                              <1>
0000CEEB F6C101               <1>       test   cl, 1 ; 1 = read, 2 = write, 3 = read&write
0000CEEE 74EF                 <1>       jz     short rw3
                              <1>
0000CEF0 89C3                 <1>       mov    ebx, eax
0000CEF2 66C1E302             <1>       shl    bx, 2 ; *4
                              <1>
0000CEF6 F6C580               <1>       test   ch, 80h ; bit 7, installable device driver flag
0000CEF9 7406                 <1>       jz     short d_read_2 ; Kernel device
                              <1>       ; installable device
                              <1> d_read_1:
0000CEFB FFA3[B0660100]       <1>         jmp dword [ebx+IDEV_RADDR-4]
                              <1> d_read_2:
0000CF01 FFA3[50150100]       <1>       jmp    dword [ebx+KDEV_RADDR-4]
                              <1>
                              <1> device_write:
                              <1>       ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
                              <1>       ; cl = DEV_OPENMODE ; open mode
                              <1>       ; ch = DEV_ACCESS   ; access flags
                              <1>       ; al = DEV_DRIVER   ; device number (eax)
                              <1>
0000CF07 F6C102               <1>       test   cl, 2 ; 1 = read, 2 = write, 3 = read&write
0000CF0A 74D3                 <1>       jz     short rw3
                              <1>
0000CF0C 89C3                 <1>       mov    ebx, eax
```

```
0000CF0E 66C1E302             <1>       shl    bx, 2 ; *4
                              <1> 
0000CF12 F6C580               <1>       test   ch, 80h ; bit 7, installable device driver flag
0000CF15 7406                 <1>       jz     short d_write_2 ; Kernel device
                              <1>       ; installable device
                              <1> d_write_1:
0000CF17 FFA3[D0660100]       <1>         jmp  dword [ebx+IDEV_WADDR-4]
                              <1> d_write_2:
0000CF1D FFA3[A0150100]       <1>       jmp    dword [ebx+KDEV_WADDR-4]
                              <1> 
                              <1> 
                              <1> sysemt: ; enable (or disable) multi tasking -time sharing-
                              <1>       ;
                              <1>       ; 23/05/2016 - TRDOS 386 (TRDOS v2.0)
                              <1>       ; 14/05/2015 (Retro UNIX 386 v1)
                              <1>       ; 10/12/2013 - 20/04/2014 (Retro UNIX 8086 v1)
                              <1>       ;
                              <1>       ; Retro UNIX 8086 v1 modification:
                              <1>       ;      'Enable Multi Tasking'  system call instead
                              <1>       ;      of 'Emulator Trap' in original UNIX v1 for PDP-11.
                              <1>       ;
                              <1>       ; Retro UNIX 8086 v1 feature only!
                              <1>       ;      Using purpose: Kernel will start without time-out
                              <1>       ;      (internal clock/timer) functionality.
                              <1>       ;      Then etc/init will enable clock/timer for
                              <1>       ;      multi tasking.
                              <1>       ;
                              <1>       ; INPUT ->
                              <1>       ;      BL = 0 -> disable multi tasking
                              <1>       ;      BL > 1 -> enable multi tasking (time sharing)
                              <1>       ; OUTPUT ->
                              <1>       ;      none
                              <1>       ;
                              <1>       ;  Note: Multi tasking is disabled during system
                              <1>       ;       initialization, it must be enabled by using
                              <1>       ;       this system call. (Otherwise, running proces
                              <1>       ;       will not be changed by another process within
                              <1>       ;       run time sequence/schedule, if running process
                              <1>       ;       will not 'release' itself. Only 'wakeup' procedure
                              <1>       ;       for waiting processes and programmed timer events
                              <1>       ;       for other processes can change running process
                              <1>       ;       while multi tasking is disabled.) ** 23/05/2016 **
                              <1> 
0000CF23 803D[B0030300]00     <1>       cmp    byte [u.uid], 0 ; root ?
                              <1>       ;ja    error
0000CF2A 0F87D3F8FFFF         <1>       ja     badsys ; 14/05/2015
                              <1>       ;
0000CF30 FA                   <1>       cli
0000CF31 881D[CE650100]       <1>       mov    [multi_tasking], bl ; 0 to disable, >0 to enable
0000CF37 E9A2F7FFFF           <1>       jmp    sysret
                              <1> 
                              <1> systimer:
                              <1>       ; 02/01/2017
                              <1>       ; 21/12/2016
                              <1>       ; 19/12/2016
                              <1>       ; 10/12/2016 (callback)
                              <1>       ; 10/06/2016
                              <1>       ; 07/06/2016
                              <1>       ; 06/06/2016
                              <1>       ; 21/05/2016
                              <1>       ; 19/05/2016
                              <1>       ; 18/05/2016 - TRDOS 386 (TRDOS v2.0)
                              <1>       ; (TRDOS 386 feature only!)
                              <1>       ;
                              <1>       ; (start or stop timer event(s))
                              <1>       ;
                              <1>       ; INPUT ->
                              <1>       ;      BL = Signal return byte (response byte)
                              <1>       ;           (Any requested value between 0 and 255)
                              <1>       ;           (Kernel will put it at the requested address)
                              <1>       ;      BH = Time count unit
                              <1>       ;           0 = Stop timer event
                              <1>       ;           1 = 18.2 ticks per second
                              <1>       ;           2 = 10 milliseconds
                              <1>       ;           3 = 1 second (for real time clock interrupt)
                              <1>       ;           4 = time/tick count in current time count unit
                              <1>       ;          // 10/12/2016
                              <1>       ;          80h = Stop timer event (callback method)
                              <1>       ;          81h = 18.2 ticks per second, callback method
                              <1>       ;          82h = 10 milliseconds, callback method
                              <1>       ;          83h = 1 second (for RTC int), callback method
                              <1>       ;          84h = current time count unit, callback method
                              <1>       ;
                              <1>       ;          Note: Only 03h or 83h will set real time clock
                              <1>       ;               (RTC) events (Others are for PIT events)!
                              <1>       ;
                              <1>       ;      NOTE: If callback (user service) method is used,
                              <1>       ;          EDX will point to the return address (of service
                              <1>       ;          procedure) in user's space instead of signal
                              <1>       ;          response byte address. (TRDOS 386 kernel will
                              <1>       ;          direct the cpu to that address -in user's space-
                              <1>       ;          at the return of system call or interrupt
                              <1>       ;          just after the adjusted count/time is elapsed.)
                              <1>       ;          User's sevice routine must be ended with a
                              <1>       ;          'iret'. Normal return addresses from system
                              <1>       ;          calls or and interrupts will be kept same except
                              <1>       ;          the timer returns.
                              <1>       ;
                              <1>       ;      BH = 0 -> Stop timer event
                              <1>       ;      BL = Timer event number (1 to 255) if BH = 0
                              <1>       ;           If BL = 0, all timer events (which are belongs
                              <1>       ;            to running process) will be stopped
                              <1>       ;      ECX = Time/Tick count (depending on time count unit)
```

```
                             <1>       ;      EDX = Signal return (Response) byte address
                             <1>       ;            (virtual address in user's memory space)
                             <1>       ; OUTPUT ->
                             <1>       ;      AL = Timer event number   (1 to 255) (max. value = 16)
                             <1>       ;      IF BH Input = 0 & CF = 0 & AL = 0 ->
                             <1>       ;           timer event(s) has/have been stopped/finished
                             <1>       ;      CF = 1 & AL = 0 -> no timer setting space to set
                             <1>       ;      CF = 1 & AL > 0 -> timer count unit is not usable
                             <1>       ;
                             <1>       ;      NOTE: To modify a time count for a user function,
                             <1>       ;            at first, current timer event must be stopped
                             <1>       ;            then a new timer event (which is related with
                             <1>       ;            same user function) must be started.
                             <1>       ;
                             <1>       ;            Signal return (response) byte may be used for
                             <1>       ;            several purposes. Kernel will put this value
                             <1>       ;            to requested address during timer interrupt,
                             <1>       ;            program/user can check this value to understand
                             <1>       ;            which event has been occurred and what is changed.
                             <1>       ;            (Multi timer events can share same signal address)
                             <1>       ;
                             <1>       ;      NOTE: If the process is running while the time count
                             <1>       ;            is reached, kernel will put signal return (response)
                             <1>       ;            byte value at requested address during timer
                             <1>       ;            interrupt and the process will continue to run.
                             <1>       ;            Program/process must call (jump to) it's timer event
                             <1>       ;            function as required, for checking the timer event
                             <1>       ;            status via signal return (response) byte address.
                             <1>       ;
                             <1>       ;            If the process is not running (waiting or sleeping
                             <1>       ;            or released) while the time count is reached,
                             <1>       ;            it is restarted from where it left, to ensure
                             <1>       ;            proper multi media (video, audio, clock, timer)
                             <1>       ;            functionality.
                             <1>       ;
                             <1>       ;            (It is better to use 'syswait' or 'syssleep',
                             <1>       ;            or 'sysrele' system call just after the timer
                             <1>       ;            function. Otherwise, timer events may block other
                             <1>       ;            processes which are not using timer events.)

                             <1>       ;
                             <1>       ; Timer Event Structure: (max. 16 timer events, 16*16 bytes)
                             <1>       ;       Owner:             resb 1 ; 0 = free
                             <1>       ;                          ;>0 = process number (u.uno)
                             <1>       ;      Calback:     resb 1 ; 1 = callback, 0 = response byte
                             <1>       ;      Interrupt:      resb 1 ; 0 = Timer interrupt (or none)
                             <1>       ;                          ; 1 = Real Time Clock interrupt
                             <1>       ;      Response:       resb 1 ; 0 to 255, signal return value
                             <1>       ;      Count Limit: resd 1 ; count of ticks (total/set)
                             <1>       ;      Current Count:      resd 1 ; count of ticks (current)
                             <1>       ;      Response Addr:  resd 1 ; response byte (pointer) address
                             <1>       ;
                             <1>
                             <1>       ; 19/12/2016 (timer callback)
0000CF3C C605[0C6B0100]00    <1>       mov    byte [tcallback], 0
0000CF43 C605[0D6B0100]00    <1>       mov    byte [trtc], 0
0000CF4A C705[D0030300]0000- <1>       mov    dword [u.tcb], 0 ; this is not necessary...
0000CF52 0000                <1>
                             <1>
0000CF54 80FF80              <1>       cmp    bh, 80h
0000CF57 7225                <1>       jb     short systimer_cb2
0000CF59 7704                <1>       ja     short systimer_cb0
                             <1>
0000CF5B 31D2                <1>       xor    edx, edx ; 0, reset callback address
0000CF5D EB0B                <1>       jmp    short systimer_cb1
                             <1>
                             <1> systimer_cb0:
0000CF5F 80FF84              <1>       cmp    bh, 84h
0000CF62 7764                <1>       ja     short systimer_5 ;  undefined, error
                             <1>
                             <1>       ;mov   byte [tcallback], 1 ; 19/12/2016
0000CF64 FE05[0C6B0100]      <1>       inc    byte [tcallback]
                             <1>
                             <1> systimer_cb1:
0000CF6A 0FB635[B3030300]    <1>       movzx  esi, byte [u.uno] ; process number
0000CF71 66C1E602            <1>       shl    si, 2
0000CF75 8996[0C010300]      <1>       mov    [esi+p.tcb-4], edx ; set process timer callback address
                             <1>                              ; (overwrite prev value if it is set!)
0000CF7B 80E77F              <1>       and    bh, 7Fh
                             <1>
                             <1> systimer_cb2:
0000CF7E 80FF02              <1>       cmp    bh, 2
0000CF81 7445                <1>         je      short systimer_5  ; only 18.2 ticks per second is usable
                             <1>                             ; 10 milliseconds (100 Hertz) timer
                             <1>                             ; will be set later (18/05/2016)
0000CF83 774B                <1>         ja      short systimer_6
                             <1>
0000CF85 20FF                <1>       and    bh, bh
0000CF87 0F84BA000000        <1>         jz      systimer_9        ; stop timer event(s)
                             <1>
                             <1>       ; bh = 1 (timer interrupt, 18.2 Hz, IBM PC/AT ROMBIOS default)
                             <1>
                             <1> systimer_19:
0000CF8D B00A                <1>       mov    al, 10 ; (*)
                             <1>
                             <1> systimer_0:
0000CF8F B710                <1>       mov    bh, 16
                             <1>       ;
0000CF91 383D[CF650100]      <1>       cmp    [timer_events], bh ; 16 ; 07/06/2016
0000CF97 7319                <1>       jnb    short systimer_3  ; max. 16 timer events
                             <1>       ;
0000CF99 50                  <1>       push   eax ; (*)
                             <1>
```

```
0000CF9A BF[60040300]        <1>       mov    edi, timer_set  ; beginning address of timer events
                             <1>                           ; setting space
0000CF9F 30C0                <1>       xor    al, al ; 0
                             <1> systimer_1:
0000CFA1 FEC0                <1>       inc    al
0000CFA3 803F00              <1>       cmp    byte [edi], 0       ; is it free space ?
0000CFA6 7639                <1>       jna    short systimer_7 ; yes
0000CFA8 FECF                <1>       dec    bh
0000CFAA 7405                <1>       jz     short systimer_2
0000CFAC 83C710              <1>       add    edi, 16
0000CFAF EBF0                <1>       jmp    short  systimer_1 ; next event space
                             <1>
                             <1> systimer_2:
0000CFB1 58                  <1>       pop    eax ; (*) discard
                             <1> systimer_3:
0000CFB2 C605[64030300]00    <1>       mov    byte [u.r0], 0
                             <1> systimer_4:
0000CFB9 C705[C8030300]1B00- <1>         mov     dword [u.error], ERR_MISC
0000CFC1 0000                <1>
                             <1>                                  ; one of miscellaneous/other errors
0000CFC3 E9F6F6FFFF          <1>       jmp    error ; cf -> 1
                             <1>
                             <1> systimer_5:
0000CFC8 883D[64030300]      <1>       mov    [u.r0], bh ; Time count unit (=2 or >3)
0000CFCE EBE9                <1>       jmp    short systimer_4 ; 07/06/2016
                             <1>
                             <1> systimer_6:
0000CFD0 80FF04              <1>       cmp    bh, 4
0000CFD3 77F3                <1>         ja      short systimer_5  ; undefined time count unit
                             <1>       ;jb    short systimer_16
                             <1>
                             <1>       ;mov   al, 1  ; default (use current timer unit)
                             <1>                    ; countdown value is in ECX !
                             <1>                    ; max. value of ecx = 4294967296/10
                             <1>         ;jmp    short systimer_0
                             <1>       ;jmp   short systimer_19
0000CFD5 74B6                <1>       je     short systimer_19
                             <1>
                             <1> systimer_16:
                             <1>       ; bh = 3
                             <1>       ; timer event via real time clock interrupt
                             <1>       ; interrupt/update frequency: 1 Hz (1 tick per second)
                             <1>
0000CFD7 B0B6                <1>       mov    al, 182 ; (*) ; 18.2 * 10
0000CFD9 FE05[0D6B0100]      <1>       inc    byte [trtc] ; timer event via real time clock
0000CFDF EBAE                <1>         jmp     short systimer_0
                             <1>
                             <1> systimer_7:
0000CFE1 A2[64030300]        <1>       mov    [u.r0], al ; timer event number
                             <1>       ;
                             <1>       ; edi = address of empty timer event area
0000CFE6 A0[B3030300]        <1>       mov    al, [u.uno]
0000CFEB FA                  <1>       cli    ; disable interrupts
0000CFEC AA                  <1>       stosb  ; process number
0000CFED A0[0C6B0100]        <1>       mov    al, [tcallback] ; timer callback flag
0000CFF2 AA                  <1>       stosb  ; 1= callback method, 0= signal response byte method
0000CFF3 A0[0D6B0100]        <1>       mov    al, [trtc] ; timer interrupt type
0000CFF8 AA                  <1>       stosb  ; 1= real time clock, 0= programmable interval timer
0000CFF9 88D8                <1>       mov    al, bl ; Signal return (Response) value
0000CFFB AA                  <1>       stosb   ; response byte
0000CFFC 58                  <1>       pop    eax ; (*) ; 10 or 182
0000CFFD 89D3                <1>       mov    ebx, edx ; virtual address for response/signal byte
0000CFFF F7E1                <1>       mul    ecx
                             <1>       ; (eax = 10 * count of 18.2 Hz timer ticks)
                             <1>       ; (count down step = 10)
0000D001 AB                  <1>       stosd  ; count limit (reset value)
0000D002 AB                  <1>       stosd  ; current count value
                             <1>
                             <1>       ; 19/12/2016
0000D003 803D[0C6B0100]00    <1>       cmp    byte [tcallback], 0 ; timer callback method ?
0000D00A 7604                <1>       jna    short systimer_17 ; no
0000D00C 89D8                <1>       mov    eax, ebx ; virtual address for callback routine
0000D00E EB0D                <1>       jmp    short systimer_18
                             <1>
                             <1> systimer_17: ; signal response byte method
                             <1>       ; ebx = virtual address
                             <1>       ; [u.pgdir] = page directory's physical address
                             <1>       ; 20/02/2017
0000D010 FE05[0E6B0100]      <1>       inc     byte [no_page_swap] ; 1
                             <1>                    ; Do not add this page to swap queue
                             <1>                    ; and remove it from swap queue if it is
                             <1>                    ; on the queue.
0000D016 E87482FFFF          <1>       call   get_physical_addr
0000D01B 721A                <1>       jc     short systimer_8 ; 07/06/2016
                             <1>       ; eax = physical address of the virtual address in user's space
                             <1> systimer_18:
0000D01D AB                  <1>       stosd  ; response addr (physical) or callback addr (virtual)
0000D01E FE05[CF650100]      <1>       inc    byte [timer_events] ; 07/06/201
                             <1>       ; 02/01/2017
0000D024 0FB605[B3030300]    <1>       movzx  eax, byte [u.uno]
0000D02B FE80[FF000300]      <1>       inc    byte [eax+p.timer-1]
                             <1>       ;
0000D031 FB                  <1>       sti    ; enable interrupts
0000D032 E9A7F6FFFF          <1>       jmp    sysret
                             <1>
                             <1> systimer_8:
                             <1>       ; 10/06/2016
                             <1>       ; 07/06/2016
0000D037 28C0                <1>       sub    al, al ; 0
0000D039 8847F4              <1>       mov    [edi-12], al ; clear process number (free timer event)
                             <1>       ;mov   dword [edi], eax ; 0
0000D03C FB                  <1>       sti
0000D03D A2[64030300]        <1>       mov    [u.r0], al ; 0
```

```
0000D042 E977F6FFFF          <1>       jmp    error
                             <1>
                             <1> systimer_9:
                             <1>       ; 10/06/2016
                             <1>       ; 07/06/2016
0000D047 28C0                <1>       sub    al, al
0000D049 A2[64030300]        <1>       mov    byte [u.r0], al ; 0
0000D04E 3805[CF650100]      <1>       cmp     byte [timer_events], al ;  0
0000D054 7631                <1>       jna    short systimer_12
                             <1>
                             <1>       ; Note: ecx and edx are undefined here
                             <1>       ;      (for stop timer function)
                             <1>
0000D056 BE[60040300]        <1>       mov    esi, timer_set  ; beginning address of timer events
                             <1>                              ; setting space
0000D05B A0[B3030300]        <1>       mov    al, [u.uno]
                             <1>
0000D060 B710                <1>       mov    bh, 16
                             <1>
0000D062 08DB                <1>       or     bl, bl
0000D064 7544                <1>       jnz    short systimer_15
                             <1>
                             <1>       ; clear timer event areas belong to current process
                             <1>       ; (for stopping all timer events belong to current process)
0000D066 FA                  <1>       cli    ; disable interrupts
                             <1> systimer_10:
                             <1>       ; 10/06/2016
                             <1>       ; 07/06/2016
0000D067 8A26                <1>       mov    ah, [esi]
0000D069 08E4                <1>       or     ah, ah ; 0 ?
0000D06B 7411                <1>       jz     short systimer_11
0000D06D 38C4                <1>       cmp    ah, al ; is the process number (owner) same ?
0000D06F 750D                <1>         jne     short systimer_11 ; no
                             <1>
                             <1>       ;mov   byte [esi], 0
0000D071 66C7060000          <1>       mov    word [esi], 0 ; clear
                             <1>       ;mov   dword [esi+12], 0 ; clear
                             <1>
0000D076 FE0D[CF650100]      <1>       dec    byte [timer_events]
0000D07C 7409                <1>       jz     short systimer_12
                             <1>
                             <1> systimer_11:
0000D07E FECF                <1>       dec    bh
0000D080 7405                <1>       jz     short systimer_12
0000D082 83C610              <1>       add    esi, 16
0000D085 EBE0                <1>       jmp    short systimer_10
                             <1>
                             <1> systimer_12:
0000D087 0FB635[B3030300]    <1>       movzx  esi, byte [u.uno]
0000D08E 08DB                <1>       or     bl, bl ; all timer events or one timer event ?
0000D090 740C                <1>       jz     short systimer_13
0000D092 8A9E[FF000300]      <1>       mov    bl, [esi+p.timer-1]
0000D098 20DB                <1>       and    bl, bl ; previous number of timer events for the process
0000D09A 7408                <1>       jz     short systimer_14
0000D09C FECB                <1>       dec    bl  ; previous number of timer events for the process - 1
                             <1> systimer_13:
0000D09E 889E[FF000300]      <1>       mov    [esi+p.timer-1], bl ; 0 ; no timer events for process
                             <1> systimer_14:
0000D0A4 FB                  <1>       sti    ; enable interrupts
0000D0A5 E934F6FFFF          <1>       jmp    sysret
                             <1>
                             <1> systimer_15:
0000D0AA 38FB                <1>       cmp    bl, bh ; 16
0000D0AC 0F8707FFFFFF        <1>         ja      systimer_4      ; max. 16 timer events !
                             <1>       ;
0000D0B2 88DA                <1>       mov    dl, bl
0000D0B4 FECA                <1>       dec    dl  ; 16 -> 15 ... 1 -> 0
0000D0B6 C0E204              <1>       shl    dl, 4 ; * 16
0000D0B9 0FB6FA              <1>       movzx  edi, dl
0000D0BC 01F7                <1>       add    edi, esi ; timer_set
                             <1>
0000D0BE 3A07                <1>       cmp    al, [edi] ; process number
0000D0C0 0F85F3FEFFFF        <1>         jne     systimer_4
                             <1>
                             <1>       ; same process ID
0000D0C6 FA                  <1>       cli    ; disable interrupts
                             <1>       ; 10/06/2016 ; 02/01/2017
                             <1>       ;mov   byte [edi], 0
0000D0C7 66C7070000          <1>       mov    word [edi], 0 ; clear
                             <1>       ;mov   dword [edi+12], 0 ; clear
0000D0CC FE0D[CF650100]      <1>       dec    byte [timer_events]
0000D0D2 EBB3                <1>       jmp    short systimer_12
                             <1>
                             <1> sysvideo: ; VIDEO DATA TRANSFER FUNCTIONS
                             <1>       ; 12/05/2017
                             <1>       ; 11/07/2016
                             <1>       ; 13/06/2016
                             <1>       ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ;
                             <1>       ;
                             <1>       ; VIDEO DATA TRANSFER FUNCTIONS:
                             <1>       ;
                             <1>       ; Inputs:
                             <1>       ;      BH = 0 = VIDEO BIOS Mode 3, tty/text mode data transfers
                             <1>       ;           BL =
                             <1>       ;             Bits 0&1, Transfer direction
                             <1>       ;                   0 - System to system
                             <1>       ;                   1 - User to system
                             <1>       ;                   2 - System to user
                             <1>       ;                   3 - User to user
                             <1>       ;             Bits 2&3, Transfer Type
                             <1>       ;                   0 - Display page transfer
                             <1>       ;                   1 - Display page window transfer
```

```
<1>       ;                  2 - Frame/Viewport/Window address transfer
<1>       ;                  3 - Window handle transfer
<1>       ;
<1>       ;          /// BL = 0 -> System to system (display page) transfer
<1>       ;             CL = Source page
<1>       ;             DL = Destination page
<1>       ;          /// BL = 1&2 -> user to system & system to user transfer
<1>       ;             ECX = User buffer
<1>       ;             DL = Video page
<1>       ;          /// BL = 5&6 -> user to system, system to user transfer
<1>       ;            (window in current display page and in current mode)
<1>       ;             ESI = User's buffer address
<1>       ;             ECX Low 16 bits = Top left column (X1 position)
<1>       ;             ECX High 16 bits = Top row (Y1 position)
<1>       ;             EDX Low 16 bits = Bottom right column (X2 position)
<1>       ;             EDX High 16 bits = Bottom row (Y2 position)
<1>         ;                If BL = 5 ->
<1>       ;             EDI = Swap address (in user's memory space)
<1>       ;             (If swap address > 0, previous content of the window
<1>       ;             will be saved into swap area in user's memory space)
<1>       ;          /// BL = 4 -> system to system transfer
<1>       ;             ESI = System's source buffer (video page) address
<1>       ;             ECX Low 16 bits = Top left column (X1 position)
<1>       ;             ECX High 16 bits = Top row (Y1 position)
<1>       ;             EDX Low 16 bits = Bottom right column (X2 position)
<1>       ;             EDX High 16 bits = Bottom row (Y2 position)
<1>       ;             EDI = System's destination buffer (video page) address
<1>       ;
<1>       ;     BH = 1 = CGA Graphics (0B8000h) data transfers
<1>       ;          BL =
<1>       ;            0 = Fill color (color in CL] (32K)
<1>       ;            1 = User to system display page transfer
<1>       ;            2 = System to user display page transfer
<1>       ;            3 = NOT bits in window (ECX, EDX)
<1>       ;            4 = Window copy (system to system)
<1>       ;            5 = User to system window transfer
<1>       ;            6 = System to user window transfer
<1>       ;            7 = AND display page bytes with CL
<1>       ;            8 = OR display page bytes with CL
<1>       ;            9 = XOR display page bytes with CL
<1>       ;
<1>       ;          /// BL = 0 -> Fill color  (all screen pixels)
<1>       ;             CL = Color value
<1>       ;          /// BL = 1&2 -> user to system & system to user transfer
<1>       ;             ECX = User buffer
<1>       ;          /// BL = 5&6 -> user to system, system to user transfer
<1>       ;            (window in current display page and in current mode)
<1>       ;             ESI = User's buffer address
<1>       ;             ECX Low 16 bits = Top left column (X1 position)
<1>       ;             ECX High 16 bits = Top row (Y1 position)
<1>       ;             EDX Low 16 bits = Bottom right column (X2 position)
<1>       ;             EDX High 16 bits = Bottom row (Y2 position)
<1>         ;        /// BL = 4 -> system to system (window) transfer
<1>       ;             ESI = System's source buffer (video page) address
<1>       ;             ECX Low 16 bits = Top left column (X1 position)
<1>       ;             ECX High 16 bits = Top row (Y1 position)
<1>       ;             EDX Low 16 bits = Bottom right column (X2 position)
<1>       ;             EDX High 16 bits = Bottom row (Y2 position)
<1>       ;             EDI = System's destination buffer (video page) address
<1>       ;          /// BL = 3 -> NOT byte in display page/memory
<1>       ;             ECX Low 16 bits = Top left column (X1 position)
<1>       ;             ECX High 16 bits = Top row (Y1 position)
<1>       ;             EDX Low 16 bits = Bottom right column (X2 position)
<1>       ;             EDX High 16 bits = Bottom row (Y2 position)
<1>       ;
<1>       ;     BH = 2 = VGA Graphics (0A0000h) data transfers
<1>       ;          BL =
<1>       ;            x0h = Fill color (color in CL] (64K)
<1>       ;            x1h = User to system display page transfer
<1>       ;            x2h = System to user display page transfer
<1>       ;            x3h = NOT bits in window (ECX, EDX)
<1>       ;            x4h = Window copy (system to system)
<1>       ;            x5h = User to system window transfer
<1>       ;            x6h = System to user window transfer
<1>       ;            x7h = AND display page bytes with CL
<1>       ;            x8h = OR display page bytes with CL
<1>       ;            x9h = XOR display page bytes with CL
<1>       ;            x = 0 -> screen width = 320
<1>       ;            x = 1 -> screen width = 640
<1>       ;            x = 2 -> screen width = 800
<1>       ;
<1>       ;          /// BL = 0 -> Fill color  (all screen pixels)
<1>       ;             CL = Color value
<1>       ;          /// BL = 1&2 -> user to system & system to user transfer
<1>       ;             ECX = User buffer
<1>       ;          /// BL = 5&6 -> user to system, system to user transfer
<1>       ;            (window in current display page and in current mode)
<1>       ;             ESI = User's buffer address
<1>       ;             ECX Low 16 bits = Top left column (X1 position)
<1>       ;             ECX High 16 bits = Top row (Y1 position)
<1>       ;             EDX Low 16 bits = Bottom right column (X2 position)
<1>       ;             EDX High 16 bits = Bottom row (Y2 position)
<1>              ;          /// BL = 4 -> system to system (window) transfer
<1>       ;             ESI = System's source buffer (video page) address
<1>       ;             ECX Low 16 bits = Top left column (X1 position)
<1>       ;             ECX High 16 bits = Top row (Y1 position)
<1>       ;             EDX Low 16 bits = Bottom right column (X2 position)
<1>       ;             EDX High 16 bits = Bottom row (Y2 position)
<1>       ;             EDI = System's destination buffer (video page) address
<1>       ;          /// BL = 3 -> NOT byte in display page/memory
<1>       ;             ECX Low 16 bits = Top left column (X1 position)
<1>       ;             ECX High 16 bits = Top row (Y1 position)
<1>       ;             EDX Low 16 bits = Bottom right column (X2 position)
```

```
<1>       ;             EDX High 16 bits = Bottom row (Y2 position)
<1>       ;
<1>       ;     BH = 3 = Super VGA, LINEAR FRAME BUFFER data transfers
<1>       ;          BL =
<1>       ;            0 = Fill color (color in ECX] (Frame buffer size)
<1>       ;            1 = User to system display page transfer
<1>       ;            2 = System to user display page transfer
<1>       ;            3 = NOT bits in window (ECX, EDX)
<1>       ;            4 = Window copy (system to system)
<1>       ;            5 = User to system window transfer
<1>       ;            6 = System to user window transfer
<1>       ;            7 = AND display page bytes with ECX
<1>       ;            8 = OR display page bytes with ECX
<1>       ;            9 = XOR display page bytes with ECX
<1>       ;
<1>       ;          /// BL = 0 -> Fill color  (all screen pixels)
<1>       ;             CL = Color value
<1>       ;          /// BL = 1&2 -> user to system & system to user transfer
<1>       ;             ECX = User buffer
<1>       ;          /// BL = 5&6 -> user to system, system to user transfer
<1>       ;            (window in current display page and in current mode)
<1>       ;             ESI = User's buffer address
<1>       ;             ECX Low 16 bits = Top left column (X1 position)
<1>       ;             ECX High 16 bits = Top row (Y1 position)
<1>       ;             EDX Low 16 bits = Bottom right column (X2 position)
<1>       ;             EDX High 16 bits = Bottom row (Y2 position)
<1>       ;          /// BL = 4 -> system to system (window) transfer
<1>       ;             ESI = System's source buffer (video page) address
<1>       ;             ECX Low 16 bits = Top left column (X1 position)
<1>       ;             ECX High 16 bits = Top row (Y1 position)
<1>       ;             EDX Low 16 bits = Bottom right column (X2 position)
<1>       ;             EDX High 16 bits = Bottom row (Y2 position)
<1>       ;             EDI = System's destination buffer (video page) address
<1>       ;          /// BL = 3 -> NOT byte in display page/memory
<1>       ;             ECX Low 16 bits = Top left column (X1 position)
<1>       ;             ECX High 16 bits = Top row (Y1 position)
<1>       ;             EDX Low 16 bits = Bottom right column (X2 position)
<1>       ;             EDX High 16 bits = Bottom row (Y2 position)
<1>       ;
<1>       ; Outputs:
<1>       ;     EAX = transfer/byte count
<1>       ;
<1>       ;     NOTE: If the source or destination address passes out of
<1>       ;     video pages (display memory limits), data will not be transferred
<1>       ;     and EAX will return as 0.
<1>       ;
<1>       ;
<1>       ; DIRECT (STANDARD VGA/CGA) DISPLAY MEMORY ACCESS FUNCTIONS:
<1>       ;
<1>       ;     BH = 4 = CGA direct video memory (0B8000h, 32K) access
<1>       ;            Page directory & page tables of the user's
<1>       ;            program will be updated to direct access to
<1>       ;            0B8000h (32K) video (CGA, color) memory; if
<1>       ;            there is not a permission  conflict or lock!
<1>       ;              (User's program/process will have permision to
<1>       ;            access locked display memory if the owner is
<1>       ;            it's parent.)
<1>         ;
<1>       ;         Screen width = 320
<1>       ;
<1>       ;     BH = 5 = VGA direct video memory (0A0000h, 64K) access
<1>       ;            Page directory & page tables of the user's
<1>       ;            program will be updated to direct access to
<1>       ;            0A0000h (64K) video (VGA) memory; if there is not
<1>       ;            a permission conflict or lock!
<1>       ;              (User's program/process will have permision to
<1>       ;            access locked display memory if the owner is
<1>       ;            it's parent.)
<1>       ;
<1>       ;         BL = Screen width (320, 640, 800)
<1>       ;
<1>       ; Outputs:
<1>       ;     EAX = Display mmory address for direct access
<1>       ;           0A0000h for VGA, 0B8000h for CGA
<1>       ;     (Display memory size: 32K for CGA, 64K for VGA)
<1>       ;     EAX = 0 if display page access permission has been denied.
<1>       ;           (Locked!)
<1>       ;
<1>       ; LINEAR FRAME BUFFER ACCESS FUNCTIONS:
<1>       ;
<1>       ;     BH = 6 = Linear Frame Buffer direct video memory access
<1>       ;
<1>       ;            Page directory & page tables of the user's
<1>       ;            program will be updated to direct access to
<1>       ;            the configured LFB (Linear Frame Buffer) address,
<1>       ;            if there is not a permission conflict or lock!
<1>       ;              (User's program/process will have permision to
<1>       ;            access locked display memory if the owner is
<1>       ;            it's parent.)
<1>       ;
<1>       ;            Return: EAX = Linear Frame Buffer address
<1>       ;                   EDX = Frame Buffer Size in bytes
<1>       ;
<1>       ;     BH = 7 = Get Linear Frame Buffer info (for current mode)
<1>       ;
<1>       ;            Return:
<1>       ;            EAX = Frame Buffer Address (0 = is not in use)
<1>       ;            EDX = Frame Buffer Size in bytes
<1>       ;            BL = Current Video Mode
<1>       ;                 BL = 0FFh -> Super VGA (Extended VGA)
<1>       ;                 If BL = 0FFh,
<1>         ;                       BH = 0 = 16 colors
<1>       ;                   BH = 1 = 256 colors
```

```
                             <1>       ;                   BH = 2 = 66536 colors
                             <1>       ;                   BH = 3 = 24 bits TRUE (16M) colors
                             <1>       ;                   BH = 4 = 32 bits TRUE (16M) colors
                             <1>       ;             ECX = Pixel resolution
                             <1>       ;                   CX = Width (640, 800, 1024, 1366, 1920)
                             <1>       ;                   High 16 bits of ECX = Height
                             <1>       ;
                             <1>       ;      NOTE: Each process will have it's own frame buffer
                             <1>       ;            address and resolution parameters in 'u' area.
                             <1>       ;            Then, if the current frame buffer & resolution
                             <1>       ;            is different, frame buffer r/w functions
                             <1>       ;            will use scale factor to convert process's
                             <1>         ;              pixel coordinates to actual screen coordinates.
                             <1>       ;            resolution -> dimensional scale
                             <1>       ;            color size -> color scale
                             <1>       ;           * RGB (TRUE) colors to 256 colors conversion:
                             <1>         ;              TRUE Colors -> 8,8,8 (R,G,B; byte 0 is R)
                             <1>       ;            256 colors -> 2,2,2,2 (R,G,B,L; bit 0&1 is R)
                             <1>       ;               bit 6&7 -> luminosity base level (0,1,2,3)
                             <1>       ;               bit 4&5 -> blue level (0,1,2,3)
                             <1>       ;               bit 2%3 -> green level (0,1,2,3)
                             <1>       ;               bit 0&1 -> red level (0,1,2,3)
                             <1>       ;            Example: total red level : luminosity + red level
                             <1>       ;            Luminosity base level: 0 -> 16
                             <1>       ;                               1 -> 32
                             <1>       ;                               2 -> 64
                             <1>       ;                               3 -> 128
                             <1>       ;            Color level:
                             <1>       ;                              0 -> 0
                             <1>       ;                              1 -> luminosity level
                             <1>       ;                              2 -> luminosity level + 64
                             <1>       ;                              3 -> 255
                             <1>       ;           Luminosity base level = min (R,G,B)
                             <1>       ;                   if it is <16, it will be set to 16
                             <1>       ;           Color levels: Color values are fixed to (nearest)
                             <1>       ;                one of all possible set level (step) values
                             <1>       ;                (according to luminosity base level); then
                             <1>       ;                color levels are set to R-L, G-L, B-L.
                             <1>       ;       For example: If luminosity base level is 32
                             <1>       ;               all possible set values are 0, 32, 96, 255.
                             <1>       ;
                             <1>       ;          * RGB (TRUE) colors to 16 colors conversion:
                             <1>       ;          16 colors: R, B,G, L bits (4 bits)
                             <1>       ;                If any one of R,G,B >= 128 L = 1
                             <1>       ;                If max. value of (R,G,B) >= 32, it is 1
                             <1>       ;                   else all color bits (R&G&B&L) are 0
                             <1>       ;                If the second value >= max. value / 2
                             <1>       ;                   it is 1
                             <1>       ;                If third value value >= max. value / 2
                             <1>       ;                   it is 1
                             <1>       ;          Example: R = 132, G = 64, B = 78
                             <1>       ;                  L = 1, R = 1
                             <1>       ;                  G < 66 --> G = 0
                             <1>       ;                  B >= 66 --> B = 1

                             <1>
                             <1>       ; 16/05/2016
0000D0D4 31C0                <1>       xor    eax, eax
0000D0D6 A3[64030300]        <1>       mov    [u.r0], eax
                             <1>
0000D0DB 20FF                <1>       and    bh, bh
0000D0DD 0F8572020000        <1>       jnz    sysvideo_13 ; 11/07/2016
                             <1>
                             <1>       ; Video mode 0, 80*25 text mode, CGA 16 colors  ; [CRT_MODE] = 3
0000D0E3 88DF                <1>       mov    bh, bl
0000D0E5 C0EF02              <1>       shr    bh, 2
0000D0E8 20FF                <1>       and    bh, bh
0000D0EA 0F8598000000        <1>         jnz      sysvideo_4
0000D0F0 BF00800B00          <1>       mov    edi, 0B8000h
0000D0F5 20D2                <1>       and    dl, dl
0000D0F7 7413                <1>       jz     short sysvideo_1
0000D0F9 80FA07              <1>       cmp    dl, 7
0000D0FC 0F87DCF5FFFF        <1>       ja     sysret
                             <1> sysvideo_0:
0000D102 81C7A00F0000        <1>       add    edi, 80*25*2
0000D108 FECA                <1>       dec    dl
0000D10A 75F6                <1>       jnz    short sysvideo_0
                             <1> sysvideo_1:
0000D10C 80E303              <1>       and    bl, 3
0000D10F 7530                <1>       jnz    short sysvideo_2
0000D111 80F907              <1>       cmp    cl, 7
0000D114 0F87C4F5FFFF        <1>       ja     sysret
                             <1>       ; system to system video/display page transfer (mode 0)
0000D11A BE00800B00          <1>       mov    esi, 0B8000h
0000D11F 0FB6C1              <1>       movzx eax, cl
0000D122 BAA00F0000          <1>       mov    edx, 80*25*2
0000D127 F7E2                <1>       mul    edx
0000D129 01C6                <1>       add    esi, eax
0000D12B B9A00F0000          <1>       mov    ecx, (80*25*2)
0000D130 890D[64030300]      <1>       mov    [u.r0], ecx
0000D136 66C1E902            <1>       shr    cx, 2 ; /4
0000D13A F3A5                <1>       rep    movsd
0000D13C E99DF5FFFF          <1>       jmp    sysret
                             <1> sysvideo_2:
0000D141 80FB02              <1>       cmp    bl, 2
0000D144 0F8794F5FFFF        <1>         ja       sysret
0000D14A 721F                <1>       jb     short sysvideo_3
                             <1>       ; system to user video/display page transfer (mode 0)
0000D14C 89FE                <1>       mov    esi, edi
0000D14E 89CF                <1>       mov    edi, ecx ; user buffer
0000D150 B9A00F0000          <1>       mov    ecx, 80*25*2
0000D155 E81F160000          <1>       call   transfer_to_user_buffer ; fast transfer
0000D15A 0F827EF5FFFF        <1>       jc     sysret
```

```
0000D160 890D[64030300]      <1>       mov    [u.r0], ecx
0000D166 E973F5FFFF          <1>       jmp    sysret
                             <1> sysvideo_3:
                             <1>       ; user to system video/display page transfer (mode 0)
0000D16B 89CE                <1>       mov    esi, ecx ; user buffer
                             <1>       ; edi = video page address
0000D16D B9A00F0000          <1>       mov    ecx, 80*25*2
0000D172 E84C160000          <1>       call   transfer_from_user_buffer ; fast transfer
0000D177 0F8261F5FFFF        <1>       jc     sysret
0000D17D 890D[64030300]      <1>       mov    [u.r0], ecx
0000D183 E956F5FFFF          <1>       jmp    sysret
                             <1> sysvideo_4:
0000D188 80E303              <1>       and    bl, 3
0000D18B 0F85F6000000        <1>         jnz     sysvideo_9
0000D191 80F907              <1>       cmp    cl, 7
0000D194 0F8744F5FFFF        <1>       ja     sysret
                             <1>       ; system to system video/display page window transfer (mode 0)
0000D19A 81FE00800B00        <1>       cmp    esi, 0B8000h
0000D1A0 0F8238F5FFFF        <1>       jb     sysret
0000D1A6 81FE00FD0B00        <1>       cmp    esi, 0B8000h+(80*25*2*8)
0000D1AC 0F832CF5FFFF        <1>       jnb    sysret
0000D1B2 81FF00800B00        <1>       cmp    edi, 0B8000h
0000D1B8 0F8220F5FFFF        <1>       jb     sysret
0000D1BE 81FF00FD0B00        <1>         cmp     edi, 0B8000h+(80*25*2*8)
0000D1C4 0F8314F5FFFF        <1>       jnb    sysret
                             <1>       ;
0000D1CA 51                  <1>       push   ecx
0000D1CB 52                  <1>       push   edx
0000D1CC 0FB7C1              <1>       movzx  eax, cx ; top left column
0000D1CF 50                  <1>       push   eax
0000D1D0 C1E910              <1>       shr    ecx, 16 ; top row
0000D1D3 66B8A000            <1>       mov    ax, 80*2 ; 80 colums, 160 bytes per row
0000D1D7 F7E1                <1>       mul    ecx
0000D1D9 01C6                <1>       add    esi, eax
0000D1DB 01C7                <1>       add    edi, eax
0000D1DD 58                  <1>       pop    eax
0000D1DE 66D1E0              <1>       shl    ax, 1 ; *2
0000D1E1 01C6                <1>       add    esi, eax
0000D1E3 01C7                <1>       add    edi, eax
0000D1E5 5A                  <1>       pop    edx
0000D1E6 59                  <1>       pop    ecx
0000D1E7 B800FD0B00          <1>       mov    eax, 0B8000h+(80*25*2*8)
0000D1EC 39C6                <1>       cmp    esi, eax
0000D1EE 0F83EAF4FFFF        <1>       jnb    sysret
0000D1F4 39C6                <1>       cmp    esi, eax
0000D1F6 0F83E2F4FFFF        <1>       jnb    sysret
                             <1>
0000D1FC 56                  <1>       push   esi ; ****
0000D1FD 57                  <1>       push   edi ; ***
0000D1FE 52                  <1>       push   edx ; **
0000D1FF 51                  <1>       push   ecx ; *
0000D200 C1E910              <1>       shr    ecx, 16 ; top row
0000D203 C1EA10              <1>       shr    edx, 16 ; bottom row
0000D206 83F918              <1>       cmp    ecx, 24 ; max. 25 rows
0000D209 7773                <1>       ja     short sysvideo_6
0000D20B 83FA18              <1>       cmp    edx, 24 ; max. 25 rows
0000D20E 776E                <1>       ja     short sysvideo_6
0000D210 28CA                <1>       sub    dl, cl
0000D212 726A                <1>       jc     short sysvideo_6
0000D214 50                  <1>       push   eax ; *****
0000D215 89D3                <1>       mov    ebx, edx ; row count - 1
0000D217 B8A0000000          <1>       mov    eax, 80*2
0000D21C F7E0                <1>       mul    eax
0000D21E 01C6                <1>       add    esi, eax
0000D220 01C7                <1>       add    edi, eax
0000D222 58                  <1>       pop    eax ; *****
0000D223 39C6                <1>       cmp    esi, eax
0000D225 7757                <1>       ja     short sysvideo_6
0000D227 39C7                <1>       cmp    edi, eax
0000D229 7753                <1>       ja     short sysvideo_6
0000D22B 59                  <1>       pop    ecx ; *
0000D22C 5A                  <1>       pop    edx ; **
0000D22D 81E1FFFF0000        <1>       and    ecx, 0FFFFh
0000D233 81E2FFFF0000        <1>       and    edx, 0FFFFh
0000D239 83F94F              <1>       cmp    ecx, 79 ; max. 80 columns
0000D23C 7742                <1>       ja     short sysvideo_7
0000D23E 83FA4F              <1>       cmp    edx, 79 ; max. 80 columns
0000D241 773D                <1>       ja     short sysvideo_7
0000D243 28CA                <1>       sub    dl, cl
0000D245 7639                <1>       jna    short sysvideo_7
                             <1>       ; edx = column count (width) - 1
0000D247 D0E2                <1>       shl    dl, 1
0000D249 01D6                <1>       add    esi, edx
0000D24B 01D7                <1>       add    edi, edx
0000D24D 39C6                <1>       cmp    esi, eax
0000D24F 772F                <1>       ja     short sysvideo_7
0000D251 39C7                <1>       cmp    edi, eax
0000D253 772B                <1>       ja     short sysvideo_7
0000D255 5F                  <1>       pop    edi ; ***
0000D256 5E                  <1>       pop    esi ; ****
0000D257 FEC3                <1>       inc    bl
0000D259 FEC2                <1>       inc    dl ; column count
0000D25B 88D7                <1>       mov    bh, dl
0000D25D D0E2                <1>       shl    dl, 1
0000D25F B8A0000000          <1>       mov    eax, 80*2
0000D264 28D0                <1>       sub    al, dl ; (80 - columns) * 2
                             <1> sysvideo_5:
0000D266 88F9                <1>       mov    cl, bh
0000D268 0115[64030300]      <1>       add    [u.r0], edx
0000D26E F366A5              <1>       rep    movsw
0000D271 01C6                <1>       add    esi, eax ; next row
0000D273 01C7                <1>       add    edi, eax ; next row
0000D275 FECB                <1>       dec    bl
```

```
0000D277 75ED                 <1>         jnz    short sysvideo_5
0000D279 E960F4FFFF           <1>         jmp    sysret
                              <1> 
                              <1> sysvideo_6:
0000D27E 59                   <1>         pop    ecx ; *
0000D27F 5A                   <1>         pop    edx ; **
                              <1> sysvideo_7:
0000D280 5F                   <1>         pop    edi ; ***
0000D281 5E                   <1>         pop    esi ; ****
0000D282 E957F4FFFF           <1>         jmp    sysret
                              <1> 
                              <1> sysvideo_9:
0000D287 80FB02               <1>         cmp    bl, 2
0000D28A 0F874EF4FFFF         <1>           ja      sysret
                              <1> 
0000D290 56                   <1>         push   esi ; ****
0000D291 57                   <1>         push   edi ; ***
0000D292 52                   <1>         push   edx ; **
0000D293 51                   <1>         push   ecx ; *
                              <1> 
0000D294 C1E910               <1>         shr    ecx, 16 ; top row
0000D297 C1EA10               <1>         shr    edx, 16 ; bottom row
0000D29A 83F918               <1>         cmp    ecx, 24 ; max. 25 rows
0000D29D 77DF                 <1>         ja     short sysvideo_6
0000D29F 83FA18               <1>         cmp    edx, 24 ; max. 25 rows
0000D2A2 77DA                 <1>         ja     short sysvideo_6
0000D2A4 28CA                 <1>         sub    dl, cl
0000D2A6 72D6                 <1>         jc     short sysvideo_6
                              <1> 
0000D2A8 88CD                 <1>         mov    ch, cl ; top row
0000D2AA 8A0D[66580100]       <1>         mov    cl, [ACTIVE_PAGE]
0000D2B0 BFA00F0000           <1>         mov    edi, 80*25*2
0000D2B5 D3E7                 <1>         shl    edi, cl
0000D2B7 81C760700B00         <1>         add    edi, 0B8000h - 80*25*2
                              <1> 
0000D2BD 88D7                 <1>         mov    bh, dl ; row count - 1
0000D2BF 88EA                 <1>         mov    dl, ch ; top row
0000D2C1 B8A0000000           <1>         mov    eax, 80*2
0000D2C6 F7E2                 <1>         mul    edx
0000D2C8 01C7                 <1>         add    edi, eax
                              <1> 
0000D2CA 59                   <1>         pop    ecx ; *
0000D2CB 5A                   <1>         pop    edx ; **
0000D2CC 81E1FFFF0000         <1>         and    ecx, 0FFFFh
0000D2D2 81E2FFFF0000         <1>         and    edx, 0FFFFh
0000D2D8 83F94F               <1>         cmp    ecx, 79 ; max. 80 columns
0000D2DB 77A3                 <1>         ja     short sysvideo_7
0000D2DD 83FA4F               <1>         cmp    edx, 79 ; max. 80 columns
0000D2E0 779E                 <1>         ja     short sysvideo_7
                              <1> 
0000D2E2 28CA                 <1>         sub    dl, cl
0000D2E4 769A                 <1>         jna    short sysvideo_7
                              <1> 
0000D2E6 0FB6C1               <1>         movzx  eax, cl ; left column
0000D2E9 D0E0                 <1>         shl    al, 1  ; column * 2
0000D2EB 01C7                 <1>         add    edi, eax
                              <1> 
0000D2ED FEC2                 <1>         inc    dl ; column count
0000D2EF D0E2                 <1>         shl    dl, 1
0000D2F1 88D1                 <1>         mov    cl, dl ; column count * 2
0000D2F3 B2A0                 <1>         mov    dl, 80*2
0000D2F5 58                   <1>         pop    eax ; *** (swap address)
0000D2F6 5E                   <1>         pop    esi ; ****
0000D2F7 FEC7                 <1>         inc    bh
                              <1> 
                              <1>         ;mov   edx, 80*2
0000D2F9 B2A0                 <1>         mov    dl, 80*2
                              <1>         ;
0000D2FB 80FB01               <1>         cmp    bl, 1
0000D2FE 7735                 <1>         ja     short sysvideo_11
                              <1> 
                              <1>         ; user to system video/display page window transfer (mode 0)
0000D300 21C0                 <1>         and    eax, eax ; swap address
0000D302 7413                 <1>         jz     short sysvideo_10 ; no window swap
                              <1>         ; save previous window content in user's buffer (swap address)
0000D304 56                   <1>         push   esi ; user buffer
0000D305 57                   <1>         push   edi ; beginning address of the window
0000D306 89FE                 <1>         mov    esi, edi
0000D308 89C7                 <1>         mov    edi, eax
0000D30A E86A140000           <1>         call   transfer_to_user_buffer ; fast transfer
0000D30F 5F                   <1>         pop    edi
0000D310 5E                   <1>         pop    esi
0000D311 0F82C7F3FFFF         <1>         jc     sysret
                              <1> sysvideo_10:
                              <1>         ; user to system video/display page window transfer (mode 0)
                              <1>         ; esi =      user buffer
0000D317 E8A7140000           <1>         call   transfer_from_user_buffer ; fast transfer
0000D31C 0F82BCF3FFFF         <1>         jc     sysret
0000D322 010D[64030300]       <1>         add    [u.r0], ecx
0000D328 01D7                 <1>         add    edi, edx ; next row
0000D32A 01CE                 <1>         add    esi, ecx
0000D32C FECF                 <1>         dec    bh
0000D32E 75E7                 <1>         jnz    short sysvideo_10
0000D330 E9A9F3FFFF           <1>         jmp    sysret
                              <1> 
                              <1> sysvideo_11:
                              <1>         ; system to user video/display page window transfer (mode 0)
0000D335 87FE                 <1>         xchg   edi, esi
                              <1> sysvideo_12:
                              <1>         ; esi = beginning address of the window
                              <1>         ; edi =      user buffer
0000D337 E83D140000           <1>         call   transfer_to_user_buffer ; fast transfer
0000D33C 0F829CF3FFFF         <1>         jc     sysret
```

```
0000D342 010D[64030300]      <1>       add    [u.r0], ecx
0000D348 01D6                <1>       add    esi, edx ; next row
0000D34A 01CF                <1>       add    edi, ecx
0000D34C FECF                <1>       dec    bh
0000D34E 75E7                <1>       jnz    short sysvideo_12
0000D350 E989F3FFFF          <1>       jmp    sysret
                             <1> 
                             <1> sysvideo_13:
0000D355 80FF01              <1>       cmp    bh, 1
0000D358 0F871F030000        <1>         ja      sysvideo_38
                             <1>       ; BH = 1 = CGA Graphics (0B8000h) data transfers
                             <1> 
0000D35E 20DB                <1>       and    bl, bl
0000D360 751A                <1>       jnz    short sysvideo_14
                             <1> 
                             <1>       ; BL = 0 = Fill color (color in CL] (32K)
                             <1> 
0000D362 88C8                <1>       mov    al, cl
0000D364 B900800000          <1>       mov    ecx, 32768
0000D369 66890D[64030300]    <1>       mov    [u.r0], cx
0000D370 BF00800B00          <1>       mov    edi, 0B8000h
0000D375 F3AB                <1>       rep    stosd
0000D377 E962F3FFFF          <1>       jmp    sysret
                             <1> 
                             <1> sysvideo_14:
0000D37C 80FB01              <1>       cmp    bl, 1
0000D37F 7723                <1>       ja     short sysvideo_16
                             <1> 
0000D381 89CE                <1>       mov    esi, ecx ; user buffer
                             <1>       ; BL = 1 = user to system video/display page transfer
                             <1> sysvideo_15:
0000D383 BF00800B00          <1>       mov    edi, 0B8000h
                             <1>       ; edi = video page address
0000D388 B900800000          <1>       mov    ecx, 32768
0000D38D E831140000          <1>       call   transfer_from_user_buffer ; fast transfer
0000D392 0F8246F3FFFF        <1>       jc     sysret ; [u.r0] = 0
0000D398 66890D[64030300]    <1>       mov    [u.r0], cx
0000D39F E93AF3FFFF          <1>       jmp    sysret
                             <1> 
                             <1> sysvideo_16:
0000D3A4 80FB02              <1>       cmp    bl, 2
0000D3A7 7723                <1>       ja     short sysvideo_18
                             <1> 
0000D3A9 89CF                <1>       mov    edi, ecx ; user buffer
                             <1>       ; BL = 2 = system to user video/display page transfer
                             <1> sysvideo_17:
0000D3AB BE00800B00          <1>       mov    esi, 0B8000h
0000D3B0 B900800000          <1>       mov    ecx, 32768
0000D3B5 E8BF130000          <1>       call   transfer_to_user_buffer ; fast transfer
0000D3BA 0F821EF3FFFF        <1>       jc     sysret ; [u.r0] = 0
0000D3C0 66890D[64030300]    <1>       mov    [u.r0], cx
0000D3C7 E912F3FFFF          <1>       jmp    sysret
                             <1> 
                             <1> sysvideo_18:
0000D3CC 80FB03              <1>       cmp    bl, 3
0000D3CF 777E                <1>       ja     short sysvideo_23
                             <1> 
                             <1>       ; BL = 3 = NOT bits in window (ECX, EDX)
                             <1> 
0000D3D1 BF00800B00          <1>       mov    edi, 0B8000h
0000D3D6 89FE                <1>       mov    esi, edi
                             <1> 
0000D3D8 39CA                <1>       cmp    edx, ecx ; bottom-right > top-left ?
0000D3DA 7716                <1>       ja     short sysvideo_20 ; window
                             <1>       ; full screen (update)
0000D3DC B900800000          <1>       mov    ecx, 32768
0000D3E1 66890D[64030300]    <1>       mov    [u.r0], cx
                             <1> sysvideo_19:
0000D3E8 F616                <1>       not    byte [esi] ; NOT operation
0000D3EA 46                  <1>       inc    esi
0000D3EB E2FB                <1>       loop   sysvideo_19
0000D3ED E9ECF2FFFF          <1>       jmp    sysret
                             <1> sysvideo_20:
0000D3F2 0FB7C2              <1>       movzx  eax, dx ; bottom right column
0000D3F5 6629C8              <1>       sub    ax, cx  ; - top left column
0000D3F8 0F82E0F2FFFF        <1>         jb      sysret ; invalid
0000D3FE 6640                <1>       inc    ax ; same column no == 1 column
0000D400 50                  <1>       push   eax ; byte count per window row
0000D401 52                  <1>       push   edx
0000D402 BB40010000          <1>       mov    ebx, 320 ; screen width
0000D407 89C8                <1>       mov    eax, ecx
0000D409 C1E810              <1>       shr    eax, 16  ; top row
0000D40C F7E3                <1>       mul    ebx
0000D40E 6689CA              <1>       mov    dx, cx ; top left column
0000D411 01D0                <1>       add    eax, edx
0000D413 01C6                <1>       add    esi, eax ; start address
0000D415 59                  <1>       pop    ecx ; edx
0000D416 89C8                <1>       mov    eax, ecx
0000D418 C1E810              <1>       shr    eax, 16 ; bottom row
0000D41B F7E3                <1>       mul    ebx
0000D41D 6689CA              <1>       mov    dx, cx ; bottom right column
0000D420 01D0                <1>       add    eax, edx
0000D422 01C7                <1>       add    edi, eax ; stop address (included)
0000D424 5A                  <1>       pop    edx ; byte count per window row
0000D425 81FFFFFF0B00        <1>       cmp    edi, 0BFFFFh
0000D42B 0F87ADF2FFFF        <1>       ja     sysret
0000D431 56                  <1>       push   esi
0000D432 4E                  <1>       dec    esi
                             <1> sysvideo_21:
0000D433 89D1                <1>       mov    ecx, edx
                             <1> sysvideo_22:
0000D435 46                  <1>       inc    esi
0000D436 F616                <1>       not    byte [esi]
```

```
0000D438 E2FB                 <1>       loop   sysvideo_22
0000D43A 01DE                 <1>       add    esi, ebx ; bytes per screen row
                              <1>       ;
0000D43C 39FE                 <1>       cmp    esi, edi ; stop address (included in loop)
0000D43E 76F3                 <1>       jna    short sysvideo_21
0000D440 5E                   <1>       pop    esi
0000D441 29F7                 <1>       sub    edi, esi
0000D443 66893D[64030300]     <1>       mov    [u.r0], di
0000D44A E98FF2FFFF           <1>       jmp    sysret
                              <1>
                              <1> sysvideo_23:
0000D44F 80FB04               <1>       cmp    bl, 4
0000D452 0F87A7000000         <1>         ja      sysvideo_26
                              <1>
                              <1>       ; BL = 4 = window copy (system to system)
                              <1>
0000D458 B800800B00           <1>       mov    eax, 0B8000h
0000D45D 39C6                 <1>       cmp    esi, eax
0000D45F 0F8279F2FFFF         <1>       jb     sysret
0000D465 39C7                 <1>       cmp    edi, eax
0000D467 0F8271F2FFFF         <1>       jb     sysret
0000D46D 6605FF7F             <1>       add    ax, 7FFFh ; 32767
0000D471 39C6                 <1>       cmp    esi, eax
0000D473 0F8765F2FFFF         <1>       ja     sysret
0000D479 39C7                 <1>       cmp    edi, eax
0000D47B 0F875DF2FFFF         <1>       ja     sysret
                              <1>
0000D481 39CA                 <1>       cmp    edx, ecx ; bottom-right > top-left ?
0000D483 7714                 <1>       ja     short sysvideo_24 ; window
                              <1>       ; full screen copy
0000D485 89C1                 <1>       mov    ecx, eax
0000D487 29F9                 <1>       sub    ecx, edi
0000D489 6641                 <1>       inc    cx
0000D48B 66890D[64030300]     <1>       mov    [u.r0], cx
0000D492 F3A4                 <1>       rep    movsb
0000D494 E945F2FFFF           <1>       jmp    sysret
                              <1> sysvideo_24:
0000D499 0FB7C2               <1>       movzx  eax, dx ; bottom right column
0000D49C 6629C8               <1>       sub    ax, cx  ; - top left column
0000D49F 0F8239F2FFFF         <1>         jb      sysret ; invalid
0000D4A5 6640                 <1>       inc    ax ; same column no == 1 column
0000D4A7 50                   <1>       push   eax ; byte count per window row
                              <1>       ;
0000D4A8 52                   <1>       push   edx
0000D4A9 BB40010000           <1>       mov    ebx, 320 ; screen width
0000D4AE 89C8                 <1>       mov    eax, ecx
0000D4B0 C1E810               <1>       shr    eax, 16      ; top row
0000D4B3 F7E3                 <1>       mul    ebx
0000D4B5 6689CA               <1>       mov    dx, cx ; top left column
0000D4B8 01D0                 <1>       add    eax, edx
0000D4BA 01C7                 <1>       add    edi, eax ; start address
0000D4BC 01C6                 <1>       add    esi, eax
0000D4BE 59                   <1>       pop    ecx ; edx
0000D4BF 89C8                 <1>       mov    eax, ecx
0000D4C1 C1E810               <1>       shr    eax, 16 ; bottom row
0000D4C4 F7E3                 <1>       mul    ebx
0000D4C6 6689CA               <1>       mov    dx, cx ; bottom right column
0000D4C9 01D0                 <1>       add    eax, edx
0000D4CB 5A                   <1>       pop    edx ; byte count per window row
0000D4CC 0500800B00           <1>       add    eax, 0B8000h
0000D4D1 3DFFFF0B00           <1>       cmp    eax, 0BFFFFh
0000D4D6 0F8702F2FFFF         <1>       ja     sysret
0000D4DC 57                   <1>       push   edi ; start address
0000D4DD 50                   <1>       push   eax ; stop address (included)
                              <1> sysvideo_25:
0000D4DE 89D1                 <1>       mov    ecx, edx
0000D4E0 F3A4                 <1>       rep    movsb
0000D4E2 4F                   <1>       dec    edi
0000D4E3 4E                   <1>       dec    esi
0000D4E4 01DF                 <1>       add    edi, ebx ; bytes per screen row
0000D4E6 01DE                 <1>       add    esi, ebx
                              <1>       ;
0000D4E8 3B3C24               <1>       cmp    edi, [esp] ; stop addr(included in loop)
0000D4EB 76F1                 <1>       jna    short sysvideo_25
0000D4ED 5B                   <1>       pop    ebx ; stop address
0000D4EE 5F                   <1>       pop    edi ; start address
0000D4EF 29FB                 <1>       sub    ebx, edi
0000D4F1 6643                 <1>       inc    bx
0000D4F3 66891D[64030300]     <1>       mov    [u.r0], bx
0000D4FA E9DFF1FFFF           <1>       jmp    sysret
                              <1>
                              <1> sysvideo_26:
0000D4FF 80FB05               <1>       cmp    bl, 5
0000D502 0F8795000000         <1>         ja      sysvideo_29
                              <1>
                              <1>       ; BL = 5 = window copy (user to system)
                              <1>
0000D508 B800800B00           <1>       mov    eax, 0B8000h
0000D50D 39C7                 <1>       cmp    edi, eax
0000D50F 0F82C9F1FFFF         <1>       jb     sysret
0000D515 6605FF7F             <1>       add    ax, 7FFFh ; 32767
0000D519 39C7                 <1>       cmp    edi, eax
0000D51B 0F87BDF1FFFF         <1>       ja     sysret
                              <1>
                              <1>       ; esi = user buffer (in user's memory space)
0000D521 39CA                 <1>       cmp    edx, ecx ; bottom-right > top-left ?
0000D523 0F865AFEFFFF         <1>         jna     sysvideo_15 ; full screen copy
                              <1>
0000D529 0FB7C2               <1>       movzx  eax, dx ; bottom right column
0000D52C 6629C8               <1>       sub    ax, cx  ; - top left column
0000D52F 0F82A9F1FFFF         <1>         jb      sysret ; invalid
0000D535 6640                 <1>       inc    ax ; same column no == 1 column
0000D537 50                   <1>       push   eax ; byte count per window row
```

```
                             <1>
0000D538 52                  <1>       push   edx
0000D539 BB40010000          <1>       mov    ebx, 320 ; screen width
0000D53E 89C8                <1>       mov    eax, ecx
0000D540 C1E810              <1>       shr    eax, 16      ; top row
0000D543 F7E3                <1>       mul    ebx
0000D545 6689CA              <1>       mov    dx, cx ; top left column
0000D548 01D0                <1>       add    eax, edx
0000D54A 01C7                <1>       add    edi, eax ; start address
0000D54C 59                  <1>       pop    ecx ; edx
0000D54D 89C8                <1>       mov    eax, ecx
0000D54F C1E810              <1>       shr    eax, 16 ; bottom row
0000D552 F7E3                <1>       mul    ebx
0000D554 6689CA              <1>       mov    dx, cx ; bottom right column
0000D557 01D0                <1>       add    eax, edx
0000D559 5A                  <1>       pop    edx ; byte count per window row
0000D55A 0500800B00          <1>       add    eax, 0B8000h
0000D55F 3DFFFF0B00          <1>       cmp    eax, 0BFFFFh
0000D564 0F8774F1FFFF        <1>       ja     sysret
0000D56A 57                  <1>       push   edi ; start address
0000D56B 50                  <1>       push   eax ; stop address (included)
                             <1> sysvideo_27:
0000D56C 89D1                <1>       mov    ecx, edx ; byte count
                             <1>       ; user to system video/display page window transfer
                             <1>       ; esi =      user buffer
0000D56E E850120000          <1>       call   transfer_from_user_buffer ; fast transfer
0000D573 7221                <1>       jc     short sysvideo_28
0000D575 010D[64030300]      <1>       add    [u.r0], ecx
0000D57B 01DF                <1>       add    edi, ebx ; next row
0000D57D 01CE                <1>       add    esi, ecx
0000D57F 3B3C24              <1>       cmp    edi, [esp] ; stop addr(included in loop)
0000D582 76E8                <1>       jna    short sysvideo_27
0000D584 5B                  <1>       pop    ebx ; stop address
0000D585 5F                  <1>       pop    edi ; start address
0000D586 29FB                <1>       sub    ebx, edi
0000D588 6643                <1>       inc    bx
0000D58A 66891D[64030300]    <1>       mov    [u.r0], bx
0000D591 E948F1FFFF          <1>       jmp    sysret
                             <1> sysvideo_28:
0000D596 58                  <1>       pop    eax
0000D597 5A                  <1>       pop    edx
0000D598 E941F1FFFF          <1>       jmp    sysret
                             <1>
                             <1> sysvideo_29:
0000D59D 80FB06              <1>       cmp    bl, 6
0000D5A0 0F8797000000        <1>         ja      sysvideo_32
                             <1>
                             <1>       ; BL = 6 = window copy (system to user)
                             <1>
0000D5A6 89F7                <1>       mov    edi, esi ; user buffer
                             <1>
0000D5A8 B800800B00          <1>       mov    eax, 0B8000h
0000D5AD 39C6                <1>       cmp    esi, eax
0000D5AF 0F8229F1FFFF        <1>       jb     sysret
0000D5B5 6605FF7F            <1>       add    ax, 7FFFh ; 32767
0000D5B9 39C6                <1>       cmp    esi, eax
0000D5BB 0F871DF1FFFF        <1>       ja     sysret
                             <1>
                             <1>       ; edi = user buffer (in user's memory space)
0000D5C1 39CA                <1>       cmp    edx, ecx ; bottom-right > top-left ?
0000D5C3 0F86E2FDFFFF        <1>         jna     sysvideo_17 ; full screen copy
                             <1>
0000D5C9 0FB7C2              <1>       movzx  eax, dx ; bottom right column
0000D5CC 6629C8              <1>       sub    ax, cx  ; - top left column
0000D5CF 0F8209F1FFFF        <1>         jb      sysret ; invalid
0000D5D5 6640                <1>       inc    ax ; same column no == 1 column
0000D5D7 50                  <1>       push   eax ; byte count per window row
                             <1>
0000D5D8 52                  <1>       push   edx
0000D5D9 BB40010000          <1>       mov    ebx, 320 ; screen width
0000D5DE 89C8                <1>       mov    eax, ecx
0000D5E0 C1E810              <1>       shr    eax, 16      ; top row
0000D5E3 F7E3                <1>       mul    ebx
0000D5E5 6689CA              <1>       mov    dx, cx ; top left column
0000D5E8 01D0                <1>       add    eax, edx
0000D5EA 01C6                <1>       add    esi, eax ; start address
0000D5EC 59                  <1>       pop    ecx ; edx
0000D5ED 89C8                <1>       mov    eax, ecx
0000D5EF C1E810              <1>       shr    eax, 16 ; bottom row
0000D5F2 F7E3                <1>       mul    ebx
0000D5F4 6689CA              <1>       mov    dx, cx ; bottom right column
0000D5F7 01D0                <1>       add    eax, edx
0000D5F9 5A                  <1>       pop    edx ; byte count per window row
0000D5FA 0500800B00          <1>       add    eax, 0B8000h
0000D5FF 3DFFFF0B00          <1>       cmp    eax, 0BFFFFh
0000D604 0F87D4F0FFFF        <1>       ja     sysret
0000D60A 56                  <1>       push   esi ; start address
0000D60B 50                  <1>       push   eax ; stop address (included)
                             <1> sysvideo_30:
0000D60C 89D1                <1>       mov    ecx, edx ; byte count
                             <1>       ; user to system video/display page window transfer
                             <1>       ; esi =      user buffer
0000D60E E866110000          <1>       call   transfer_to_user_buffer ; fast transfer
0000D613 7221                <1>       jc     short sysvideo_31
0000D615 010D[64030300]      <1>       add    [u.r0], ecx
0000D61B 01DF                <1>       add    edi, ebx ; next row
0000D61D 01CE                <1>       add    esi, ecx
0000D61F 3B3C24              <1>       cmp    edi, [esp] ; stop addr(included in loop)
0000D622 76E8                <1>       jna    short sysvideo_30
0000D624 5B                  <1>       pop    ebx ; stop address
0000D625 5F                  <1>       pop    edi ; start address
0000D626 29FB                <1>       sub    ebx, edi
0000D628 6643                <1>       inc    bx
```

```
0000D62A 66891D[64030300]    <1>       mov    [u.r0], bx
0000D631 E9A8F0FFFF          <1>       jmp    sysret
                             <1> sysvideo_31:
0000D636 58                  <1>       pop    eax
0000D637 5A                  <1>       pop    edx
0000D638 E9A1F0FFFF          <1>       jmp    sysret
                             <1>
                             <1> sysvideo_32:
0000D63D 80FB07              <1>       cmp    bl, 7
0000D640 770F                <1>       ja     short sysvideo_34
                             <1>
                             <1>       ; BL = 7 = AND display page bytes with CL
                             <1>
0000D642 BE00800B00          <1>       mov    esi, 0B8000h
0000D647 B900800000          <1>       mov    ecx, 32768
                             <1> sysvideo_33:
0000D64C 200E                <1>       and    byte [esi], cl
0000D64E 46                  <1>       inc    esi
0000D64F E2FB                <1>       loop   sysvideo_33
                             <1>
                             <1> sysvideo_34:
0000D651 80FB08              <1>       cmp    bl, 8
0000D654 770F                <1>       ja     short sysvideo_36
                             <1>
                             <1>       ; BL = 8 = OR display page bytes with CL
                             <1>
0000D656 BE00800B00          <1>       mov    esi, 0B8000h
0000D65B B900800000          <1>       mov    ecx, 32768
                             <1> sysvideo_35:
0000D660 080E                <1>       or     byte [esi], cl
0000D662 46                  <1>       inc    esi
0000D663 E2FB                <1>       loop   sysvideo_35
                             <1>
                             <1> sysvideo_36:
0000D665 80FB09              <1>       cmp    bl, 9
0000D668 0F8770F0FFFF        <1>         ja      sysret ; nothing to do
                             <1>
                             <1>       ; BL = 9 = XOR display page bytes with CL
                             <1>
0000D66E BE00800B00          <1>       mov    esi, 0B8000h
0000D673 B900800000          <1>       mov    ecx, 32768
                             <1> sysvideo_37:
0000D678 300E                <1>       xor    byte [esi], cl
0000D67A 46                  <1>       inc    esi
0000D67B E2FB                <1>       loop   sysvideo_37
                             <1>
                             <1> sysvideo_38:
0000D67D 80FF02              <1>       cmp    bh, 2
0000D680 0F8733030000        <1>         ja      sysvideo_64
                             <1>       ; BH = 2 = VGA Graphics (0A0000h) data transfers
                             <1>
0000D686 88DC                <1>       mov    ah, bl
0000D688 80E30F              <1>       and    bl, 0Fh
0000D68B C0EC04              <1>       shr    ah, 4
0000D68E C1E310              <1>       shl    ebx, 16
0000D691 66BB4001            <1>       mov    bx, 320 ; 320*200, 320*240
0000D695 20E4                <1>       and    ah, ah
0000D697 7413                <1>       jz     short sysvideo_39
0000D699 66D1E3              <1>       shl    bx, 1 ; 640*200, 640 * 400, 640*480
0000D69C 80FC02              <1>       cmp    ah, 2
0000D69F 720B                <1>       jb     short sysvideo_39
0000D6A1 0F8737F0FFFF        <1>       ja     sysret ; invalid
                             <1>       ; 800*600
0000D6A7 6681C3A000          <1>       add    bx, 160 ; 800
                             <1> sysvideo_39:
0000D6AC C1CB10              <1>       ror    ebx, 16
                             <1>
0000D6AF 20DB                <1>       and    bl, bl
0000D6B1 7519                <1>       jnz    short sysvideo_40
                             <1>
                             <1>       ; BL = 0 = Fill color (color in CL] (64K)
                             <1>
0000D6B3 88C8                <1>       mov    al, cl
0000D6B5 B900000100          <1>       mov    ecx, 65536
0000D6BA 890D[64030300]      <1>       mov    [u.r0], ecx
0000D6C0 BF00000A00          <1>       mov    edi, 0A0000h
0000D6C5 F3AB                <1>       rep    stosd
0000D6C7 E912F0FFFF          <1>       jmp    sysret
                             <1>
                             <1> sysvideo_40:
0000D6CC 80FB01              <1>       cmp    bl, 1
0000D6CF 7722                <1>       ja     short sysvideo_42
                             <1>
0000D6D1 89CE                <1>       mov    esi, ecx ; user buffer
                             <1>       ; BL = 1 = user to system video/display page transfer
                             <1> sysvideo_41:
0000D6D3 BF00000A00          <1>       mov    edi, 0A0000h
                             <1>       ; edi = video page address
0000D6D8 B900000100          <1>       mov    ecx, 65536
0000D6DD E8E1100000          <1>       call   transfer_from_user_buffer ; fast transfer
0000D6E2 0F82F6EFFFFF        <1>       jc     sysret ; [u.r0] = 0
0000D6E8 890D[64030300]      <1>       mov    [u.r0], ecx
0000D6EE E9EBEFFFFF          <1>       jmp    sysret
                             <1>
                             <1> sysvideo_42:
0000D6F3 80FB02              <1>       cmp    bl, 2
0000D6F6 7722                <1>       ja     short sysvideo_44
                             <1>
0000D6F8 89CF                <1>       mov    edi, ecx ; user buffer
                             <1>       ; BL = 2 = system to user video/display page transfer
                             <1> sysvideo_43:
0000D6FA BE00000A00          <1>       mov    esi, 0A0000h
0000D6FF B900000100          <1>       mov    ecx, 65536
```

```
0000D704 E870100000           <1>        call   transfer_to_user_buffer ; fast transfer
0000D709 0F82CFEFFFFF         <1>        jc     sysret ; [u.r0] = 0
0000D70F 890D[64030300]       <1>        mov    [u.r0], ecx
0000D715 E9C4EFFFFF           <1>        jmp    sysret
                              <1> 
                              <1> sysvideo_44:
0000D71A 80FB03               <1>        cmp    bl, 3
0000D71D 777A                 <1>        ja     short sysvideo_49
                              <1> 
                              <1>        ; BL = 3 = NOT bits in window (ECX, EDX)
                              <1> 
0000D71F BF00000A00           <1>        mov    edi, 0A0000h
0000D724 89FE                 <1>        mov    esi, edi
                              <1> 
0000D726 39CA                 <1>        cmp    edx, ecx ; bottom-right > top-left ?
0000D728 770B                 <1>        ja     short sysvideo_45 ; window
                              <1>        ; full screen (update)
0000D72A B900000100           <1>        mov    ecx, 65536
0000D72F 890D[64030300]       <1>        mov    [u.r0], ecx
                              <1> sysvideo_45:
0000D735 F616                 <1>        not    byte [esi] ; NOT operation
0000D737 46                   <1>        inc    esi
0000D738 E2FB                 <1>        loop   sysvideo_45
0000D73A E99FEFFFFF           <1>        jmp    sysret
                              <1> sysvideo_46:
0000D73F 0FB7C2               <1>        movzx  eax, dx ; bottom right column
0000D742 6629C8               <1>        sub    ax, cx  ; - top left column
0000D745 0F8293EFFFFF         <1>          jb      sysret ; invalid
0000D74B 6640                 <1>        inc    ax ; same column no == 1 column
0000D74D 50                   <1>        push   eax ; byte count per window row
0000D74E 52                   <1>        push   edx
0000D74F C1EB10               <1>        shr    ebx, 16 ; 320,640,800 : screen width
0000D752 89C8                 <1>        mov    eax, ecx
0000D754 C1E810               <1>        shr    eax, 16  ; top row
0000D757 F7E3                 <1>        mul    ebx
0000D759 6689CA               <1>        mov    dx, cx ; top left column
0000D75C 01D0                 <1>        add    eax, edx
0000D75E 01C6                 <1>        add    esi, eax ; start address
0000D760 59                   <1>        pop    ecx ; edx
0000D761 89C8                 <1>        mov    eax, ecx
0000D763 C1E810               <1>        shr    eax, 16 ; bottom row
0000D766 F7E3                 <1>        mul    ebx
0000D768 6689CA               <1>        mov    dx, cx ; bottom right column
0000D76B 01D0                 <1>        add    eax, edx
0000D76D 01C7                 <1>        add    edi, eax ; stop address (included)
0000D76F 5A                   <1>        pop    edx ; byte count per window row
0000D770 81FFFFFF0A00         <1>        cmp    edi, 0AFFFFh
0000D776 0F8762EFFFFF         <1>        ja     sysret
0000D77C 56                   <1>        push   esi
0000D77D 4E                   <1>        dec    esi
                              <1> sysvideo_47:
0000D77E 89D1                 <1>        mov    ecx, edx
                              <1> sysvideo_48:
0000D780 46                   <1>        inc    esi
0000D781 F616                 <1>        not    byte [esi]
0000D783 E2FB                 <1>        loop   sysvideo_48
0000D785 01DE                 <1>        add    esi, ebx ; bytes per screen row
                              <1>        ;
0000D787 39FE                 <1>        cmp    esi, edi ; stop address (included in loop)
0000D789 76F3                 <1>        jna    short sysvideo_47
0000D78B 5E                   <1>        pop    esi
0000D78C 29F7                 <1>        sub    edi, esi
0000D78E 893D[64030300]       <1>        mov    [u.r0], edi
0000D794 E945EFFFFF           <1>        jmp    sysret
                              <1> 
                              <1> sysvideo_49:
0000D799 80FB04               <1>        cmp    bl, 4
0000D79C 0F87A1000000         <1>          ja      sysvideo_52
                              <1> 
                              <1>        ; BL = 4 = window copy (system to system)
                              <1> 
0000D7A2 B800000A00           <1>        mov    eax, 0A0000h
0000D7A7 39C6                 <1>        cmp    esi, eax
0000D7A9 0F822FEFFFFF         <1>        jb     sysret
0000D7AF 39C7                 <1>        cmp    edi, eax
0000D7B1 0F8227EFFFFF         <1>        jb     sysret
0000D7B7 6683C0FF             <1>        add    ax, 0FFFFh ; 65535
0000D7BB 39C6                 <1>        cmp    esi, eax
0000D7BD 0F871BEFFFFF         <1>        ja     sysret
0000D7C3 39C7                 <1>        cmp    edi, eax
0000D7C5 0F8713EFFFFF         <1>        ja     sysret
                              <1> 
0000D7CB 39CA                 <1>        cmp    edx, ecx ; bottom-right > top-left ?
0000D7CD 7712                 <1>        ja     short sysvideo_50 ; window
                              <1>        ; full screen copy
0000D7CF 89C1                 <1>        mov    ecx, eax
0000D7D1 29F9                 <1>        sub    ecx, edi
0000D7D3 41                   <1>        inc    ecx
0000D7D4 890D[64030300]       <1>        mov    [u.r0], ecx
0000D7DA F3A4                 <1>        rep    movsb
0000D7DC E9FDEEFFFF           <1>        jmp    sysret
                              <1> sysvideo_50:
0000D7E1 0FB7C2               <1>        movzx  eax, dx ; bottom right column
0000D7E4 6629C8               <1>        sub    ax, cx  ; - top left column
0000D7E7 0F82F1EEFFFF         <1>          jb      sysret ; invalid
0000D7ED 6640                 <1>        inc    ax ; same column no == 1 column
0000D7EF 50                   <1>        push   eax ; byte count per window row
                              <1>        ;
0000D7F0 52                   <1>        push   edx
0000D7F1 C1EB10               <1>        shr    ebx, 16 ; 320,640,800 : screen width
0000D7F4 89C8                 <1>        mov    eax, ecx
0000D7F6 C1E810               <1>        shr    eax, 16      ; top row
0000D7F9 F7E3                 <1>        mul    ebx
```

```
0000D7FB 6689CA              <1>        mov    dx, cx ; top left column
0000D7FE 01D0                <1>        add    eax, edx
0000D800 01C7                <1>        add    edi, eax ; start address
0000D802 01C6                <1>        add    esi, eax
0000D804 59                  <1>        pop    ecx ; edx
0000D805 89C8                <1>        mov    eax, ecx
0000D807 C1E810              <1>        shr    eax, 16 ; bottom row
0000D80A F7E3                <1>        mul    ebx
0000D80C 6689CA              <1>        mov    dx, cx ; bottom right column
0000D80F 01D0                <1>        add    eax, edx
0000D811 5A                  <1>        pop    edx ; byte count per window row
0000D812 0500000A00          <1>        add    eax, 0A0000h
0000D817 3DFFFF0A00          <1>        cmp    eax, 0AFFFFh
0000D81C 0F87BCEEFFFF        <1>        ja     sysret
0000D822 57                  <1>        push   edi ; start address
0000D823 50                  <1>        push   eax ; stop address (included)
                             <1> sysvideo_51:
0000D824 89D1                <1>        mov    ecx, edx
0000D826 F3A4                <1>        rep    movsb
0000D828 4F                  <1>        dec    edi
0000D829 4E                  <1>        dec    esi
0000D82A 01DF                <1>        add    edi, ebx ; bytes per screen row
0000D82C 01DE                <1>        add    esi, ebx
                             <1>        ;
0000D82E 3B3C24              <1>        cmp    edi, [esp] ; stop addr(included in loop)
0000D831 76F1                <1>        jna    short sysvideo_51
0000D833 5B                  <1>        pop    ebx ; stop address
0000D834 5F                  <1>        pop    edi ; start address
0000D835 29FB                <1>        sub    ebx, edi
0000D837 43                  <1>        inc    ebx
0000D838 891D[64030300]      <1>        mov    [u.r0], ebx
0000D83E E99BEEFFFF          <1>        jmp    sysret
                             <1>
                             <1> sysvideo_52:
0000D843 80FB05              <1>        cmp    bl, 5
0000D846 0F8791000000        <1>          ja      sysvideo_55
                             <1>
                             <1>        ; BL = 5 = window copy (user to system)
                             <1>
0000D84C B800000A00          <1>        mov    eax, 0A0000h
0000D851 39C7                <1>        cmp    edi, eax
0000D853 0F8285EEFFFF        <1>        jb     sysret
0000D859 6683C0FF            <1>        add    ax, 0FFFFh ; 65535
0000D85D 39C7                <1>        cmp    edi, eax
0000D85F 0F8779EEFFFF        <1>        ja     sysret
                             <1>
                             <1>        ; esi = user buffer (in user's memory space)
0000D865 39CA                <1>        cmp    edx, ecx ; bottom-right > top-left ?
0000D867 0F8666FEFFFF        <1>          jna     sysvideo_41 ; full screen copy
                             <1>
0000D86D 0FB7C2              <1>        movzx  eax, dx ; bottom right column
0000D870 6629C8              <1>        sub    ax, cx  ; - top left column
0000D873 0F8265EEFFFF        <1>          jb      sysret ; invalid
0000D879 6640                <1>        inc    ax ; same column no == 1 column
0000D87B 50                  <1>        push   eax ; byte count per window row
                             <1>
0000D87C 52                  <1>        push   edx
0000D87D C1EB10              <1>        shr    ebx, 16 ; 320,640,800 : screen width
0000D880 89C8                <1>        mov    eax, ecx
0000D882 C1E810              <1>        shr    eax, 16      ; top row
0000D885 F7E3                <1>        mul    ebx
0000D887 6689CA              <1>        mov    dx, cx ; top left column
0000D88A 01D0                <1>        add    eax, edx
0000D88C 01C7                <1>        add    edi, eax ; start address
0000D88E 59                  <1>        pop    ecx ; edx
0000D88F 89C8                <1>        mov    eax, ecx
0000D891 C1E810              <1>        shr    eax, 16 ; bottom row
0000D894 F7E3                <1>        mul    ebx
0000D896 6689CA              <1>        mov    dx, cx ; bottom right column
0000D899 01D0                <1>        add    eax, edx
0000D89B 5A                  <1>        pop    edx ; byte count per window row
0000D89C 0500000A00          <1>        add    eax, 0A0000h
0000D8A1 3DFFFF0A00          <1>        cmp    eax, 0AFFFFh
0000D8A6 0F8732EEFFFF        <1>        ja     sysret
0000D8AC 57                  <1>        push   edi ; start address
0000D8AD 50                  <1>        push   eax ; stop address (included)
                             <1> sysvideo_53:
0000D8AE 89D1                <1>        mov    ecx, edx ; byte count
                             <1>        ; user to system video/display page window transfer
                             <1>        ; esi =      user buffer
0000D8B0 E80E0F0000          <1>        call   transfer_from_user_buffer ; fast transfer
0000D8B5 721F                <1>        jc     short sysvideo_54
0000D8B7 010D[64030300]      <1>        add    [u.r0], ecx
0000D8BD 01DF                <1>        add    edi, ebx ; next row
0000D8BF 01CE                <1>        add    esi, ecx
0000D8C1 3B3C24              <1>        cmp    edi, [esp] ; stop addr(included in loop)
0000D8C4 76E8                <1>        jna    short sysvideo_53
0000D8C6 5B                  <1>        pop    ebx ; stop address
0000D8C7 5F                  <1>        pop    edi ; start address
0000D8C8 29FB                <1>        sub    ebx, edi
0000D8CA 43                  <1>        inc    ebx
0000D8CB 891D[64030300]      <1>        mov    [u.r0], ebx
0000D8D1 E908EEFFFF          <1>        jmp    sysret
                             <1> sysvideo_54:
0000D8D6 58                  <1>        pop    eax
0000D8D7 5A                  <1>        pop    edx
0000D8D8 E901EEFFFF          <1>        jmp    sysret
                             <1>
                             <1> sysvideo_55:
0000D8DD 80FB06              <1>        cmp    bl, 6
0000D8E0 0F8793000000        <1>          ja      sysvideo_58
                             <1>
                             <1>        ; BL = 6 = window copy (system to user)
```

```
                                  <1>
0000D8E6 89F7                     <1>         mov    edi, esi ; user buffer
                                  <1>
0000D8E8 B800000A00               <1>         mov    eax, 0A0000h
0000D8ED 39C6                     <1>         cmp    esi, eax
0000D8EF 0F82E9EDFFFF             <1>         jb     sysret
0000D8F5 6683C0FF                 <1>         add    ax, 0FFFFh ; 65535
0000D8F9 39C6                     <1>         cmp    esi, eax
0000D8FB 0F87DDEDFFFF             <1>         ja     sysret
                                  <1>
                                  <1>         ; edi = user buffer (in user's memory space)
0000D901 39CA                     <1>         cmp    edx, ecx ; bottom-right > top-left ?
0000D903 0F86A2FAFFFF             <1>           jna     sysvideo_17 ; full screen copy
                                  <1>
0000D909 0FB7C2                   <1>         movzx  eax, dx ; bottom right column
0000D90C 6629C8                   <1>         sub    ax, cx  ; - top left column
0000D90F 0F82C9EDFFFF             <1>           jb      sysret ; invalid
0000D915 6640                     <1>         inc    ax ; same column no == 1 column
0000D917 50                       <1>         push   eax ; byte count per window row
                                  <1>
0000D918 52                       <1>         push   edx
0000D919 C1EB10                   <1>         shr    ebx, 16 ; 320, 640,800 ; screen width
0000D91C 89C8                     <1>         mov    eax, ecx
0000D91E C1E810                   <1>         shr    eax, 16      ; top row
0000D921 F7E3                     <1>         mul    ebx
0000D923 6689CA                   <1>         mov    dx, cx ; top left column
0000D926 01D0                     <1>         add    eax, edx
0000D928 01C6                     <1>         add    esi, eax ; start address
0000D92A 59                       <1>         pop    ecx ; edx
0000D92B 89C8                     <1>         mov    eax, ecx
0000D92D C1E810                   <1>         shr    eax, 16 ; bottom row
0000D930 F7E3                     <1>         mul    ebx
0000D932 6689CA                   <1>         mov    dx, cx ; bottom right column
0000D935 01D0                     <1>         add    eax, edx
0000D937 5A                       <1>         pop    edx ; byte count per window row
0000D938 0500000A00               <1>         add    eax, 0A0000h
0000D93D 3DFFFF0A00               <1>         cmp    eax, 0AFFFFh
0000D942 0F8796EDFFFF             <1>         ja     sysret
0000D948 56                       <1>         push   esi ; start address
0000D949 50                       <1>         push   eax ; stop address (included)
                                  <1> sysvideo_56:
0000D94A 89D1                     <1>         mov    ecx, edx ; byte count
                                  <1>         ; user to system video/display page window transfer
                                  <1>         ; esi =       user buffer
0000D94C E8280E0000               <1>         call   transfer_to_user_buffer ; fast transfer
0000D951 721F                     <1>         jc     short sysvideo_57
0000D953 010D[64030300]           <1>         add    [u.r0], ecx
0000D959 01DF                     <1>         add    edi, ebx ; next row
0000D95B 01CE                     <1>         add    esi, ecx
0000D95D 3B3C24                   <1>         cmp    edi, [esp] ; stop addr(included in loop)
0000D960 76E8                     <1>         jna    short sysvideo_56
0000D962 5B                       <1>         pop    ebx ; stop address
0000D963 5F                       <1>         pop    edi ; start address
0000D964 29FB                     <1>         sub    ebx, edi
0000D966 43                       <1>         inc    ebx
0000D967 891D[64030300]           <1>         mov    [u.r0], ebx
0000D96D E96CEDFFFF               <1>         jmp    sysret
                                  <1> sysvideo_57:
0000D972 58                       <1>         pop    eax
0000D973 5A                       <1>         pop    edx
0000D974 E965EDFFFF               <1>         jmp    sysret
                                  <1>
                                  <1> sysvideo_58:
0000D979 80FB07                   <1>         cmp    bl, 7
0000D97C 770F                     <1>         ja     short sysvideo_60
                                  <1>
                                  <1>         ; BL = 7 = AND display page bytes with CL
                                  <1>
0000D97E BE00000A00               <1>         mov    esi, 0A0000h
0000D983 B900000100               <1>         mov    ecx, 65536
                                  <1> sysvideo_59:
0000D988 200E                     <1>         and    byte [esi], cl
0000D98A 46                       <1>         inc    esi
0000D98B E2FB                     <1>         loop   sysvideo_59
                                  <1>
                                  <1> sysvideo_60:
0000D98D 80FB08                   <1>         cmp    bl, 8
0000D990 770F                     <1>         ja     short sysvideo_62
                                  <1>
                                  <1>         ; BL = 8 = OR display page bytes with CL
                                  <1>
0000D992 BE00000A00               <1>         mov    esi, 0A0000h
0000D997 B900000100               <1>         mov    ecx, 65536
                                  <1> sysvideo_61:
0000D99C 080E                     <1>         or     byte [esi], cl
0000D99E 46                       <1>         inc    esi
0000D99F E2FB                     <1>         loop   sysvideo_61
                                  <1>
                                  <1> sysvideo_62:
0000D9A1 80FB09                   <1>         cmp    bl, 9
0000D9A4 0F8734EDFFFF             <1>           ja      sysret ; nothing to do
                                  <1>
                                  <1>         ; BL = 9 = XOR display page bytes with CL
                                  <1>
0000D9AA BE00000A00               <1>         mov    esi, 0A0000h
0000D9AF B900000100               <1>         mov    ecx, 65536
                                  <1> sysvideo_63:
0000D9B4 300E                     <1>         xor    byte [esi], cl
0000D9B6 46                       <1>         inc    esi
0000D9B7 E2FB                     <1>         loop   sysvideo_63
                                  <1>
                                  <1> sysvideo_64:
0000D9B9 80FF03                   <1>         cmp    bh, 3
```

```
0000D9BC 7464                 <1>       je     short sysvideo_68
0000D9BE 80FF04               <1>       cmp    bh, 4
0000D9C1 7721                 <1>       ja     short sysvideo_65
                              <1> 
                              <1>       ; BH = 4
                              <1>       ; Direct User Access for CGA video memory.
                              <1>       ; Setup user's page tables for direct access to 0B8000h.
                              <1>       ;
                              <1>       ; Permission checks are not implemented yet !
                              <1>       ; (11/07/2016)
                              <1> 
0000D9C3 B800800B00           <1>       mov    eax, 0B8000h
0000D9C8 B908000000           <1>       mov    ecx, 8 ; 8 pages (8*4K=32K)
0000D9CD 89C3                 <1>       mov    ebx, eax ; 12/05/2017 ; virtual = physical
0000D9CF E8CD7CFFFF           <1>       call   direct_memory_access
0000D9D4 0F8204EDFFFF         <1>       jc     sysret
                              <1>       ; eax = 0B8000h if there is not an error
0000D9DA A3[64030300]         <1>       mov    [u.r0], eax
0000D9DF E9FAECFFFF           <1>       jmp    sysret
                              <1> 
                              <1> sysvideo_65:
0000D9E4 80FF05               <1>       cmp    bh, 5
0000D9E7 7721                 <1>       ja     short sysvideo_66
                              <1> 
                              <1>       ; BH = 5
                              <1>       ; Direct User Access for VGA video memory.
                              <1>       ; Setup user's page tables for direct access to 0A0000h.
                              <1>       ;
                              <1>       ; Permission checks are not implemented yet !
                              <1>       ; (11/07/2016)
                              <1> 
0000D9E9 B800000A00           <1>       mov    eax, 0A0000h
0000D9EE B910000000           <1>       mov    ecx, 16 ; 16 pages (16*4K=64K)
0000D9F3 89C3                 <1>       mov    ebx, eax ; 12/05/2017 ; virtual = physical
0000D9F5 E8A77CFFFF           <1>       call   direct_memory_access
0000D9FA 0F82DEECFFFF         <1>       jc     sysret
                              <1>       ; eax = 0A0000h if there is not an error
0000DA00 A3[64030300]         <1>       mov    [u.r0], eax
0000DA05 E9D4ECFFFF           <1>       jmp    sysret
                              <1> 
                              <1> sysvideo_66:
0000DA0A 80FF06               <1>       cmp    bh, 6
0000DA0D 7705                 <1>       ja     short sysvideo_67
                              <1>       ; BH = 6
                              <1>       ; Direct User Access for (Super VGA) Linear Frame Buffer.
                              <1>       ; Setup user's page tables for direct access to LFB.
                              <1>       ;
                              <1>       ; Not implemented yet !
                              <1>       ; (11/07/2016)
0000DA0F E9CAECFFFF           <1>       jmp    sysret
                              <1> 
                              <1> sysvideo_67:
0000DA14 80FF07               <1>       cmp    bh, 7
0000DA17 0F87C1ECFFFF         <1>       ja     sysret ; invalid !
                              <1> 
                              <1>       ; BH = 7
                              <1>       ; Get (Super/Extended VGA) Linear Frame Buffer info.
                              <1>       ;
                              <1>       ; Not implemented yet !
                              <1>       ; (11/07/2016)
0000DA1D E9BCECFFFF           <1>       jmp    sysret
                              <1> 
                              <1> sysvideo_68:
                              <1>       ; BH = 3
                              <1>       ; Super VGA, LINEAR FRAME BUFFER data transfers
                              <1>       ; Not implemented for yet ! (11/07/2016)
0000DA22 E9B7ECFFFF           <1>       jmp    sysret
                              <1> 
                              <1> mkdir:
                              <1>       ; 04/12/2015 (14 byte directory names)
                              <1>       ; 12/10/2015
                              <1>       ; 17/06/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>       ; 29/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
                              <1>       ;
                              <1>       ; 'mkdir' makes a directory entry from the name pointed to
                              <1>       ; by u.namep into the current directory.
                              <1>       ;
                              <1>       ; INPUTS ->
                              <1>       ;    u.namep - points to a file name
                              <1>       ;                 that is about to be a directory entry.
                              <1>       ;    ii - current directory's i-number.
                              <1>       ; OUTPUTS ->
                              <1>       ;    u.dirbuf+2 - u.dirbuf+10 - contains file name.
                              <1>       ;    u.off - points to entry to be filled
                              <1>       ;            in the current directory
                              <1>       ;    u.base - points to start of u.dirbuf.
                              <1>       ;    r1 - contains i-number of current directory
                              <1>       ;
                              <1>       ; ((AX = R1)) output
                              <1>       ;
                              <1>       ;    (Retro UNIX Prototype : 11/11/2012, UNIXCOPY.ASM)
                              <1>         ;    ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
                              <1>       ;
                              <1> 
                              <1>       ; 17/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
0000DA27 31C0                 <1>       xor    eax, eax
0000DA29 BF[9A030300]         <1>       mov     edi, u.dirbuf+2
0000DA2E 89FE                 <1>       mov    esi, edi
0000DA30 AB                   <1>       stosd
0000DA31 AB                   <1>       stosd
                              <1>       ; 04/12/2015 (14 byte directory names)
0000DA32 AB                   <1>       stosd
0000DA33 66AB                 <1>       stosw
```

```
                             <1>              ; jsr r0,copyz; u.dirbuf+2; u.dirbuf+10. / clear this
0000DA35 89F7                <1>       mov    edi, esi ; offset to u.dirbuf
                             <1>       ; 12/10/2015 ([u.namep] -> ebp)
                             <1>       ;mov   ebp, [u.namep]
0000DA37 E80D030000          <1>       call   trans_addr_nmbp ; convert virtual address to physical
                             <1>              ; esi = physical address (page start + offset)
                             <1>              ; ecx = byte count in the page (1 - 4096)
                             <1>       ; edi = offset to u.dirbuf (edi is not modified in trans_addr_nm)
                             <1>              ; mov u.namep,r2 / r2 points to name of directory entry
                             <1>              ; mov $u.dirbuf+2,r3 / r3 points to u.dirbuf+2
                             <1> mkdir_1: ; 1:
0000DA3C 45                  <1>       inc    ebp ; 12/10/2015
                             <1>       ;
                             <1>       ; / put characters in the directory name in u.dirbuf+2 - u.dirbuf+10
                             <1>        ; 01/08/2013
0000DA3D AC                  <1>       lodsb
                             <1>              ; movb (r2)+,r1 / move character in name to r1
0000DA3E 20C0                <1>       and    al, al
0000DA40 7427                <1>       jz     short mkdir_3
                             <1>              ; beq 1f / if null, done
0000DA42 3C2F                <1>       cmp    al, '/'
                             <1>              ; cmp r1,$'/ / is it a "/"?
0000DA44 7414                <1>       je     short mkdir_err
                             <1>       ;je    error
                             <1>              ; beq error9 / yes, error
                             <1>       ; 12/10/2015
0000DA46 6649                <1>       dec    cx
0000DA48 7505                <1>       jnz    short mkdir_2
                             <1>       ; 12/10/2015 ([u.namep] -> ebp)
0000DA4A E800030000          <1>       call   trans_addr_nm ; convert virtual address to physical
                             <1>              ; esi = physical address (page start + offset)
                             <1>              ; ecx = byte count in the page
                             <1>       ; edi = offset to u.dirbuf (edi is not modified in trans_addr_nm)
                             <1> mkdir_2:
0000DA4F 81FF[A8030300]      <1>       cmp     edi, u.dirbuf+16 ; ; 04/12/2015 (10 -> 16)
                             <1>              ; cmp r3,$u.dirbuf+10. / have we reached the last slot for
                             <1>                                   ; / a char?
0000DA55 74E5                <1>       je     short mkdir_1
                             <1>              ; beq 1b / yes, go back
0000DA57 AA                  <1>       stosb
                             <1>              ; movb r1,(r3)+ / no, put the char in the u.dirbuf
0000DA58 EBE2                <1>       jmp    short mkdir_1
                             <1>              ; br 1b / get next char
                             <1> mkdir_err:
                             <1>       ; 17/06/2015
0000DA5A C705[C8030300]1300- <1>       mov    dword [u.error], ERR_NOT_DIR ; 'not a valid directory !'
0000DA62 0000                <1>
0000DA64 E955ECFFFF          <1>       jmp    error
                             <1>
                             <1> mkdir_3: ; 1:
0000DA69 A1[78030300]        <1>       mov    eax, [u.dirp]
0000DA6E A3[80030300]        <1>       mov    [u.off], eax
                             <1>              ; mov u.dirp,u.off / pointer to empty current directory
                             <1>                            ; / slot to u.off
                             <1> wdir: ; 29/04/2013
0000DA73 C705[84030300]-     <1>         mov     dword [u.base], u.dirbuf
0000DA79 [98030300]          <1>
                             <1>              ; mov $u.dirbuf,u.base / u.base points to created file name
0000DA7D C705[88030300]1000- <1>         mov     dword [u.count], 16 ; 04/12/2015 (10 -> 16)
0000DA85 0000                <1>
                             <1>              ; mov $10.,u.count / u.count = 10
0000DA87 66A1[51040300]      <1>       mov    ax, [ii]
                             <1>              ; mov ii,r1 / r1 has i-number of current directory
0000DA8D B201                <1>       mov    dl, 1 ; owner flag mask ; RETRO UNIX 8086 v1 modification !
0000DA8F E8331D0000          <1>       call   access
                             <1>              ; jsr r0,access; 1 / get i-node and set its file up
                             <1>                            ; / for writing
                             <1>       ; AX = i-number of current directory
                             <1>       ; 01/08/2013
0000DA94 FE05[C6030300]      <1>       inc     byte [u.kcall] ; the caller is 'mkdir' sign
0000DA9A E83C0F0000          <1>       call   writei
                             <1>              ; jsr r0,writei / write into directory
0000DA9F C3                  <1>       retn
                             <1>              ; rts r0
                             <1>
                             <1> sysexec:
                             <1>       ; 18/11/2017
                             <1>       ; 14/11/2017
                             <1>       ; 13/11/2017
                             <1>       ; 24/10/2016, 04/01/2017
                             <1>       ; 24/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 23/06/2015 - 23/10/2015 (Retro UNIX 386 v1)
                             <1>       ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'sysexec' initiates execution of a file whose path name if
                             <1>       ; pointed to by 'name' in the sysexec call.
                             <1>       ; 'sysexec' performs the following operations:
                             <1>       ;    1. obtains i-number of file to be executed via 'namei'.
                             <1>       ;    2. obtains i-node of file to be exceuted via 'iget'.
                             <1>       ;    3. sets trap vectors to system routines.
                             <1>       ;    4. loads arguments to be passed to executing file into
                             <1>       ;      highest locations of user's core
                             <1>       ;    5. puts pointers to arguments in locations immediately
                             <1>       ;      following arguments.
                             <1>       ;    6.       saves number of arguments in next location.
                             <1>       ;    7. intializes user's stack area so that all registers
                             <1>       ;      will be zeroed and the PS is cleared and the PC set
                             <1>       ;      to core when 'sysret' restores registers
                             <1>       ;      and does an rti.
                             <1>       ;    8. inializes u.r0 and u.sp
                             <1>       ;    9. zeros user's core down to u.r0
                             <1>       ;   10.       reads executable file from storage device into core
                             <1>       ;      starting at location 'core'.
```

```
                              <1>       ;   11.      sets u.break to point to end of user's code with
                              <1>       ;     data area appended.
                              <1>       ;   12.      calls 'sysret' which returns control at location
                              <1>       ;     'core' via 'rti' instruction.
                              <1>       ;
                              <1>       ; Calling sequence:
                              <1>       ;     sysexec; namep; argp
                              <1>       ; Arguments:
                              <1>       ;     namep - points to pathname of file to be executed
                              <1>       ;     argp  - address of table of argument pointers
                              <1>       ;     argp1... argpn - table of argument pointers
                              <1>       ;     argp1:<...0> ... argpn:<...0> - argument strings
                              <1>       ; Inputs: (arguments)
                              <1>       ; Outputs: -
                              <1>       ; ..........................................................
                              <1>       ;
                              <1>       ; Retro UNIX 386 v1 modification:
                              <1>       ;     User application runs in it's own virtual space
                              <1>       ;     which is izolated from kernel memory (and other
                              <1>       ;     memory pages) via 80386   paging in ring 3
                              <1>       ;     privilige mode. Virtual start address is always 0.
                              <1>       ;     User's core memory starts at linear address 400000h
                              <1>       ;     (the end of the 1st 4MB).
                              <1>       ;
                              <1>       ; Retro UNIX 8086 v1 modification:
                              <1>       ;     user/application segment and system/kernel segment
                              <1>       ;     are different and sysenter/sysret/sysrele routines
                              <1>       ;     are different (user's registers are saved to
                              <1>       ;     and then restored from system's stack.)
                              <1>       ;
                              <1>       ;     NOTE: Retro UNIX 8086 v1 'arg2' routine gets these
                              <1>       ;           arguments which were in these registers;
                              <1>       ;           but, it returns by putting the 1st argument
                              <1>       ;           in 'u.namep' and the 2nd argument
                              <1>       ;           on top of stack. (1st argument is offset of the
                              <1>       ;           file/path name in the user's program segment.)
                              <1>
                              <1>       ;call arg2
                              <1>       ; * name - 'u.namep' points to address of file/path name
                              <1>       ;          in the user's program segment ('u.segmnt')
                              <1>       ;          with offset in BX register (as sysopen argument 1).
                              <1>       ; * argp - sysexec argument 2 is in CX register
                              <1>       ;          which is on top of stack.
                              <1>       ;
                              <1>             ; jsr r0,arg2 / arg0 in u.namep,arg1 on top of stack
                              <1>
                              <1>       ; 23/06/2015 (32 bit modifications)
                              <1>
                              <1>       ;; 13/11/2017
                              <1>       ;;mov  [u.namep], ebx ; argument 1
                              <1>         ; 18/10/2015
0000DAA0 890D[4C040300]       <1>       mov     [argv], ecx  ; * ; argument 2
                              <1>
                              <1>       ; 13/11/2017
0000DAA6 89DE                 <1>       mov    esi, ebx
0000DAA8 E84E210000           <1>       call   set_working_path_x
0000DAAD 7319                 <1>       jnc    short sysexec_0
                              <1>
                              <1>       ;; 'bad command or file name'
                              <1>       ;mov   eax, ERR_BAD_CMD_ARG ; 01h ; TRDOS 8086
                              <1>
                              <1>       ; 'file not found !' error
0000DAAF B802000000           <1>       mov    eax, ERR_NOT_FOUND ; 02h ; TRDOS 8086
                              <1> sysexec_not_found_err:
                              <1> sysexec_access_error:
                              <1> sysexec_ext_error:
0000DAB4 A3[64030300]         <1>       mov    [u.r0], eax
0000DAB9 A3[C8030300]         <1>       mov    [u.error], eax
0000DABE E80D220000           <1>       call   reset_working_path
0000DAC3 E9F6EBFFFF           <1>       jmp    error
                              <1>
                              <1> sysexec_0:
                              <1>       ; 13/11/2017
                              <1>       ;mov   esi, FindFile_Name
0000DAC8 66B80018             <1>         mov ax, 1800h ; Only files
0000DACC E892A7FFFF           <1>       call   find_first_file
0000DAD1 72E1                 <1>       jc     short sysexec_not_found_err ; eax = 2
                              <1>
                              <1>       ; check_ file attributes
                              <1>       ; (attribute bits = 00ADVSHR) ; 18h = Directory+Volume
                              <1>       ; BL = Attributes byte
                              <1>
0000DAD3 F6C306               <1>         test bl, 6  ; system file or hidden file (S+H)
                              <1>       ;jz    short sysexec_0ext
0000DAD6 7417                 <1>       jz     short sysexec_1 ; yes
                              <1>
                              <1>       ; 13/11/2017
                              <1>       ; /// TRDOS386 permission check for multiuser mode ///
                              <1>       ; SYSTEM file or HIDDEN file !!
                              <1>       ; (Only super user has permission to run this file.)
                              <1>
                              <1>       ; ([u.uid]=0 for super user or root in multiuser mode)
                              <1>       ; ([u.uid]=0 for any users in singleuser mode)
0000DAD8 803D[B0030300]00     <1>       cmp    byte [u.uid], 0 ; Super User ([u.uid]=0) ?
                              <1>       ;jna   short sysexec_0ext
0000DADF 760E                 <1>       jna    short sysexec_1 ; yes
                              <1>
                              <1>       ; 'permission denied !' error
0000DAE1 B80B000000           <1>         mov  eax, ERR_FILE_ACCESS  ; 11 = ERR_PERM_DENIED
0000DAE6 EBCC                 <1>         jmp  short sysexec_access_error
                              <1>
                              <1> sysexec_not_exf:
                              <1>       ; 'not executable file !' error
```

```
0000DAE8 B816000000           <1>       mov    eax, ERR_NOT_EXECUTABLE
0000DAED EBC5                 <1>       jmp    sysexec_ext_error
                              <1>
                              <1> ;sysexec_0ext:
                              <1> sysexec_1:
                              <1>       ; 18/11/2017
0000DAEF BE[E4620100]         <1>       mov    esi, FindFile_Name
                              <1>       ; 13/11/2017
                              <1>       ; check program file name extension
                              <1>       ; ('.PRG' for current TRDOS version)
0000DAF4 E80DC2FFFF           <1>       call   check_prg_filename_ext
0000DAF9 72ED                 <1>       jc     short sysexec_not_exf
                              <1>
                              <1>       ; 18/11/2017
0000DAFB 3C50                 <1>       cmp    al, 'P'
0000DAFD 75E9                 <1>       jne    short sysexec_not_exf
                              <1>
                              <1>       ; '.PRG' extension is OK.
                              <1>       ; Only '.PRG' files are valid program files
                              <1>       ; for current TRDOS 386 version.
                              <1>
0000DAFF 8B15[10630100]       <1>       mov    edx, [FindFile_DirEntry+DirEntry_FileSize]
0000DB05 66A1[08630100]       <1>       mov    ax, [FindFile_DirEntry+DirEntry_FstClusHI]
0000DB0B C1E010               <1>       shl    eax, 16
0000DB0E 66A1[0E630100]       <1>       mov    ax, [FindFile_DirEntry+DirEntry_FstClusLO]
                              <1>       ; EAX = First Cluster number
                              <1>       ; EDX = File Size
                              <1>
0000DB14 A3[51040300]         <1>       mov    [ii], eax
0000DB19 8915[55040300]       <1>       mov    [i.size], edx
                              <1>
                              <1> ;sysexec_1:
                              <1>       ; 13/11/2017 - TRDOS 386 (TRDOS v2.0)
                              <1>       ; 24/06/2015 - 23/10/2015 (Retro UNIX 386 v1)
                              <1>         ; Moving arguments to the end of [u.upage]
                              <1>       ; (by regarding page borders in user's memory space)
                              <1>       ;
                              <1>       ; 10/10/2015
                              <1>       ; 21/07/2015
0000DB1F 89E5                 <1>       mov    ebp, esp ; (**)
                              <1>       ; 18/10/2015
0000DB21 89EF                 <1>       mov    edi, ebp
0000DB23 B900010000           <1>       mov    ecx, MAX_ARG_LEN ; 256
                              <1>       ;sub   edi, MAX_ARG_LEN ; 256
0000DB28 29CF                 <1>       sub    edi, ecx
0000DB2A 89FC                 <1>       mov    esp, edi ; *!*
0000DB2C 31C0                 <1>       xor    eax, eax
0000DB2E A3[8C030300]         <1>       mov    [u.nread], eax ; 0
0000DB33 66A3[4A040300]       <1>       mov    [argc], ax ; 0 ; 13/11/2017
0000DB39 49                   <1>       dec    ecx ; 256 - 1
0000DB3A 890D[88030300]       <1>       mov    [u.count], ecx ; MAX_ARG_LEN - 1 ; 255
                              <1>       ;mov   dword [u.count], MAX_ARG_LEN - 1 ; 255
                              <1> sysexec_2:
0000DB40 8B35[4C040300]       <1>       mov    esi, [argv] ; 18/10/2015
0000DB46 E866000000           <1>       call   get_argp
0000DB4B B904000000           <1>       mov    ecx, 4 ; mov ecx, 4
                              <1> sysexec_3:
0000DB50 21C0                 <1>       and    eax, eax
0000DB52 0F8429050000         <1>         jz      sysexec_6
                              <1>       ; 18/10/2015
0000DB58 010D[4C040300]       <1>       add    [argv], ecx ; 4
0000DB5E 66FF05[4A040300]     <1>       inc    word [argc]
                              <1>       ;
0000DB65 A3[84030300]         <1>       mov    [u.base], eax
                              <1>       ; 23/10/2015
0000DB6A 66C705[C4030300]00-  <1>       mov    word [u.pcount], 0
0000DB72 00                   <1>
                              <1> sysexec_4:
0000DB73 E8A10B0000           <1>       call   cpass ; get a character from user's core memory
0000DB78 750E                 <1>         jnz  short sysexec_5
                              <1>              ; (max. 255 chars + null)
                              <1>       ; 18/10/2015
0000DB7A 28C0                 <1>       sub    al, al
0000DB7C AA                   <1>       stosb
0000DB7D FF05[8C030300]       <1>       inc    dword [u.nread]
0000DB83 E9F9040000           <1>       jmp    sysexec_6 ; 24/04/2016
                              <1> sysexec_5:
0000DB88 AA                   <1>       stosb
0000DB89 20C0                 <1>       and    al, al
0000DB8B 75E6                 <1>       jnz    short sysexec_4
0000DB8D B904000000           <1>       mov    ecx, 4
0000DB92 390D[48040300]       <1>       cmp    [ncount], ecx ; 4
0000DB98 72A6                 <1>       jb     short sysexec_2
0000DB9A 8B35[44040300]       <1>       mov    esi, [nbase]
0000DBA0 010D[44040300]       <1>       add    [nbase], ecx ; 4
0000DBA6 66290D[48040300]     <1>       sub    [ncount], cx
0000DBAD 8B06                 <1>       mov    eax, [esi]
0000DBAF EB9F                 <1>       jmp    short sysexec_3
                              <1>
                              <1> get_argp:
                              <1>       ; 14/11/2017 - TRDOS 386 (TRDOS v2.0)
                              <1>       ; 18/10/2015 (nbase, ncount)
                              <1>       ; 21/07/2015
                              <1>       ; 24/06/2015 (Retro UNIX 386 v1)
                              <1>       ; Get (virtual) address of argument from user's core memory
                              <1>       ;
                              <1>       ; INPUT:
                              <1>       ;      esi = virtual address of argument pointer
                              <1>       ; OUTPUT:
                              <1>       ;      eax = virtual address of argument
                              <1>       ;
                              <1>       ; Modified registers: EAX, EBX, ECX, EDX, ESI
                              <1>       ;
```

```
0000DBB1 833D[BC030300]00    <1>       cmp     dword [u.ppgdir], 0 ; /etc/init ?
                             <1>                               ; (the caller is kernel)
0000DBB8 7667                <1>         jna     short get_argpk
                             <1>       ;
0000DBBA 89F3                <1>       mov    ebx, esi
0000DBBC E8CE76FFFF          <1>       call   get_physical_addr ; get physical address
0000DBC1 0F8289000000        <1>         jc      get_argp_err
0000DBC7 A3[44040300]        <1>       mov    [nbase], eax ; physical address
0000DBCC 66890D[48040300]    <1>       mov    [ncount], cx ; remain byte count in page (1-4096)
0000DBD3 B804000000          <1>       mov    eax, 4 ; 21/07/2015
0000DBD8 6639C1              <1>       cmp    cx, ax ; 4
0000DBDB 735D                <1>       jnb    short get_argp2
0000DBDD 89F3                <1>       mov    ebx, esi
0000DBDF 01CB                <1>       add    ebx, ecx
0000DBE1 E8A976FFFF          <1>       call   get_physical_addr ; get physical address
0000DBE6 7268                <1>       jc     short get_argp_err
                             <1>       ;push  esi
0000DBE8 89C6                <1>       mov    esi, eax
0000DBEA 66870D[48040300]    <1>       xchg   cx, [ncount]
0000DBF1 8735[44040300]      <1>       xchg   esi, [nbase]
0000DBF7 B504                <1>       mov    ch, 4
0000DBF9 28CD                <1>       sub    ch, cl
                             <1> get_argp0:
0000DBFB AC                  <1>       lodsb
0000DBFC 6650                <1>       push   ax
0000DBFE FEC9                <1>       dec    cl
0000DC00 75F9                <1>         jnz     short get_argp0
0000DC02 8B35[44040300]      <1>       mov    esi, [nbase]
                             <1>       ; 21/07/2015
0000DC08 0FB6C5              <1>       movzx  eax, ch
0000DC0B 0105[44040300]      <1>       add    [nbase], eax
0000DC11 662905[48040300]    <1>       sub    [ncount], ax
                             <1> get_argp1:
0000DC18 AC                  <1>       lodsb
0000DC19 FECD                <1>       dec    ch
0000DC1B 7447                <1>         jz      short get_argp3
0000DC1D 6650                <1>         push ax
0000DC1F EBF7                <1>       jmp      short get_argp1
                             <1> get_argpk:
                             <1>       ; Argument is in kernel's memory space
0000DC21 66C705[48040300]00- <1>       mov    word [ncount], PAGE_SIZE ; 4096
0000DC29 10                  <1>
0000DC2A 8935[44040300]      <1>       mov    [nbase], esi
0000DC30 8305[44040300]04    <1>       add    dword [nbase], 4
0000DC37 8B06                <1>       mov    eax, [esi] ; virtual addr. = physcal addr.
0000DC39 C3                  <1>       retn
                             <1> get_argp2:
                             <1>       ; 21/07/2015
                             <1>       ;mov   eax, 4
0000DC3A 8B15[44040300]      <1>       mov    edx, [nbase] ; 18/10/2015
0000DC40 0105[44040300]      <1>       add    [nbase], eax
0000DC46 662905[48040300]    <1>       sub    [ncount], ax
                             <1>       ;
0000DC4D 8B02                <1>       mov    eax, [edx]
0000DC4F C3                  <1>       retn
                             <1> get_argp_err:
0000DC50 A3[C8030300]        <1>       mov    [u.error], eax
                             <1>       ; 14/11/2017
0000DC55 B801000000          <1>       mov    eax, ERR_BAD_CMD_ARG ; 01h ; TRDOS 8086
0000DC5A A3[64030300]        <1>       mov    [u.r0], eax
0000DC5F E95AEAFFFF          <1>       jmp    error
                             <1> get_argp3:
0000DC64 B103                <1>       mov    cl, 3
                             <1> get_argp4:
0000DC66 C1E008              <1>       shl    eax, 8
0000DC69 665A                <1>       pop    dx
0000DC6B 88D0                <1>       mov    al, dl
0000DC6D E2F7                <1>         loop    get_argp4
                             <1>       ;pop   esi
0000DC6F C3                  <1>       retn
                             <1>
                             <1> sysstat:
                             <1>       ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; temporary !
0000DC70 B801000000          <1>       mov    eax, ERR_INV_FNUMBER ; 'invalid function number !'
0000DC75 A3[C8030300]        <1>         mov     [u.error], eax
0000DC7A A3[64030300]        <1>         mov     [u.r0], eax
0000DC7F E93AEAFFFF          <1>       jmp    error
                             <1>
                             <1> sysfstat:
                             <1>       ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; temporary !
0000DC84 B801000000          <1>       mov    eax, ERR_INV_FNUMBER ; 'invalid function number !'
0000DC89 A3[C8030300]        <1>         mov     [u.error], eax
0000DC8E A3[64030300]        <1>         mov     [u.r0], eax
0000DC93 E926EAFFFF          <1>       jmp    error
                             <1>
                             <1> fclose:
                             <1>       ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>       ; 18/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ;            (32 bit offset pointer modification)
                             <1>       ; 19/04/2013 - 12/01/2014 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; Given the file descriptor (index to the u.fp list)
                             <1>       ; 'fclose' first gets the i-number of the file via 'getf'.
                             <1>       ; If i-node is active (i-number > 0) the entry in
                             <1>       ; u.fp list is cleared. If all the processes that opened
                             <1>       ; that file close it, then fsp etry is freed and the file
                             <1>       ; is closed. If not a return is taken.
                             <1>       ; If the file has been deleted while open, 'anyi' is called
                             <1>       ; to see anyone else has it open, i.e., see if it is appears
                             <1>       ; in another entry in the fsp table. Upon return from 'anyi'
                             <1>       ; a check is made to see if the file is special.
                             <1>       ;
                             <1>       ; INPUTS ->
                             <1>       ;    r1 - contains the file descriptor (value=0,1,2...)
                             <1>       ;    u.fp - list of entries in the fsp table
                             <1>       ;    fsp - table of entries (4 words/entry) of open files.
                             <1>       ; OUTPUTS ->
                             <1>       ;    r1 - contains the same file descriptor
                             <1>       ;    r2 - contains i-number
                             <1>       ;
                             <1>       ; ((AX = R1))
                             <1>       ; ((Modified registers: eDX, eBX, eCX, eSI, eDI, eBP))
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 modification : CF = 1
                             <1>       ;              if i-number of the file is 0. (error)
                             <1>       ;
                             <1>       ; TRDOS 386 (06/10/2016)
                             <1>       ;
                             <1>       ; INPUT:
                             <1>       ;      EAX = File Handle (File Descriptor, File Index)
                             <1>       ;
                             <1>       ; OUTPUT:
                             <1>       ;      CF = 1 -> File not open !
                             <1>       ;      CF = 0 -> OK!
                             <1>       ;           EBX = File Number (System)
                             <1>       ;           [cdev] = Logical DOS Drive Number
                             <1>       ;           EAX = File Handle/Number (user)
                             <1>       ;
                             <1>       ; Modified Registers: EBX
                             <1>
0000DC98 50                  <1>       push   eax ; File handle
                             <1>
0000DC99 E846000000          <1>       call   getf
0000DC9E 0F8207240000        <1>       jc     device_close ; eax = device number
                             <1>
0000DCA4 80BB[62690100]01    <1>       cmp    byte [ebx+OF_MODE], 1 ; open mode ; 0 = empty entry
0000DCAB 722E                <1>       jb     short fclose_1            ; 1 = read, 2 = write
                             <1>
0000DCAD 83F801              <1>       cmp    eax, 1 ; is the first cluster number > 0
0000DCB0 7229                <1>       jb     short fclose_1 ; no, this is empty entry
                             <1>
                             <1> fclose_0:
0000DCB2 FE8B[76690100]      <1>       dec    byte [ebx+OF_OPENCOUNT] ; decrement the number of processes
                             <1>                                    ; that have opened the file
0000DCB8 7921                <1>       jns    short fclose_1 ; jump if not negative (jump if bit 7 is 0)
                             <1>                    ; if all processes haven't closed the file, return
                             <1>       ;
                             <1>       ; eax ; First cluster
0000DCBA 31C0                <1>       xor    eax, eax ; 0
0000DCBC 8883[62690100]      <1>       mov    [ebx+OF_MODE], al ; 0 = empty entry
                             <1>       ;mov   [ebx+OF_STATUS], al ; 0 = empty entry
0000DCC2 66C1E302            <1>       shl    bx, 2
0000DCC6 8983[30690100]      <1>       mov    [ebx+OF_FCLUSTER], eax ; 0
0000DCCC 8983[486A0100]      <1>       mov    [ebx+OF_CCLUSTER], eax ; 0
                             <1>       ;mov   [ebx+OF_CCINDEX], eax ; 0
0000DCD2 A3[74030300]        <1>       mov    [u.fofp], eax ; 0
0000DCD7 66C1EB02            <1>       shr    bx, 2
                             <1> fclose_1: ; 1:
0000DCDB 58                  <1>       pop    eax ; File handle (File Descriptor, File Index)
0000DCDC C680[6A030300]00    <1>       mov    byte [eax+u.fp], 0 ; clear that entry in the u.fp list
0000DCE3 C3                  <1>       retn
                             <1>
                             <1> getf:
                             <1>       ; 12/10/2016
                             <1>       ; 11/10/2016
                             <1>       ; 08/10/2016
                             <1>       ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; / get the device number and the i-number of an open file
                             <1>       ; 13/05/2015
                             <1>       ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 19/04/2013 - 18/11/2013 (Retro UNIX 8086 v1)
                             <1>       ;
0000DCE4 89C3                <1>       mov    ebx, eax
                             <1> getf1:
0000DCE6 83FB0A              <1>       cmp    ebx, 10
0000DCE9 730A                <1>         jnb short getf2
0000DCEB 8A9B[6A030300]      <1>       mov    bl, [ebx+u.fp]
0000DCF1 08DB                <1>       or     bl, bl
0000DCF3 7503                <1>       jnz    short getf3
                             <1> getf2:
                             <1>       ; 'File not open !' error (ax=0)
0000DCF5 29C0                <1>       sub    eax, eax
0000DCF7 C3                  <1>       retn
                             <1> getf3:
0000DCF8 F6C380              <1>       test   bl, 80h
0000DCFB 7530                <1>       jnz    short getf5 ; device
0000DCFD FECB                <1>       dec    bl ; 0 based
0000DCFF 8A83[58690100]      <1>       mov    al, [ebx+OF_DRIVE]
0000DD05 A2[46030300]        <1>       mov    [cdev], al
0000DD0A C0E302              <1>       shl    bl, 2 ; *4 (dword offset)
0000DD0D 8B83[A8690100]      <1>       mov    eax, [ebx+OF_SIZE]
0000DD13 A3[55040300]        <1>       mov    [i.size], eax ; file size
0000DD18 8D83[80690100]      <1>       lea    eax, [ebx+OF_POINTER] ;12/10/2016
0000DD1E A3[74030300]        <1>       mov    [u.fofp], eax
0000DD23 8B83[30690100]      <1>       mov    eax, [ebx+OF_FCLUSTER]
0000DD29 C0EB02              <1>       shr    bl, 2 ; /4 (byte offset)
                             <1> getf4:
0000DD2C C3                  <1>       retn
                             <1> getf5:
                             <1>       ; get device number
0000DD2D 80E37F              <1>       and    bl, 7Fh ; 1 to 7Fh
0000DD30 FECB                <1>       dec    bl ; 0 based (0 to 7Eh)
0000DD32 8A83[8A670100]      <1>       mov    al, [ebx+DEV_DRIVER]
```

```
0000DD38 8AAB[F4660100]      <1>        mov    ch, [ebx+DEV_ACCESS]
0000DD3E 8A8B[A8670100]      <1>        mov    cl, [ebx+DEV_OPENMODE]
0000DD44 80E5FE              <1>        and    ch, 0FEh ; reset bit 0 ; dev_close
0000DD47 F9                  <1>        stc ; cf = 1
0000DD48 C3                  <1>        retn
                             <1>
                             <1> trans_addr_nmbp:
                             <1>        ; 18/10/2015
                             <1>        ; 12/10/2015
0000DD49 8B2D[7C030300]      <1>        mov    ebp, [u.namep]
                             <1> trans_addr_nm:
                             <1>        ; Convert virtual (pathname) address to physical address
                             <1>        ; (Retro UNIX 386 v1 feature only !)
                             <1>        ; 18/10/2015
                             <1>        ; 12/10/2015 (u.pnbase & u.pncount has been removed from code)
                             <1>        ; 02/07/2015
                             <1>        ; 17/06/2015
                             <1>        ; 16/06/2015
                             <1>        ;
                             <1>        ; INPUTS:
                             <1>        ;      ebp = pathname address (virtual) ; [u.namep]
                             <1>        ;      [u.pgdir] = user's page directory
                             <1>        ; OUTPUT:
                             <1>        ;       esi = physical address of the pathname
                             <1>        ;      ecx = remain byte count in the page
                             <1>        ;
                             <1>        ; (Modified registers: EAX, EBX, ECX, EDX, ESI)
                             <1>        ;
0000DD4F 833D[BC030300]00    <1>          cmp     dword [u.ppgdir], 0  ; /etc/init ? (sysexec)
0000DD56 7618                <1>        jna    short trans_addr_nmk ; the caller is os kernel;
                             <1>                                 ; it is already physical address
0000DD58 50                  <1>        push   eax
0000DD59 89EB                <1>        mov    ebx, ebp ; [u.namep] ; pathname address (virtual)
0000DD5B E82F75FFFF          <1>               call   get_physical_addr ; get physical address
0000DD60 7204                <1>        jc     short tr_addr_nm_err
                             <1>        ; 18/10/2015
                             <1>        ; eax = physical address
                             <1>        ; cx = remain byte count in page (1-4096)
                             <1>               ; 12/10/2015 (cx = [u.pncount])
0000DD62 89C6                <1>        mov    esi, eax ; 12/10/2015 (esi=[u.pnbase])
0000DD64 58                  <1>        pop    eax
0000DD65 C3                  <1>        retn
                             <1>
                             <1> tr_addr_nm_err:
0000DD66 A3[C8030300]        <1>        mov    [u.error], eax
                             <1>        ;pop   eax
0000DD6B E94EE9FFFF          <1>        jmp    error
                             <1>
                             <1> trans_addr_nmk:
                             <1>        ; 12/10/2015
                             <1>        ; 02/07/2015
0000DD70 8B35[7C030300]      <1>        mov    esi, [u.namep]  ; [u.pnbase]
0000DD76 66B90010            <1>        mov    cx, PAGE_SIZE ; 4096 ; [u.pncount]
0000DD7A C3                  <1>        retn
                             <1>
                             <1>
                             <1> sysbreak:
                             <1>        ; 18/10/2015
                             <1>        ; 07/10/2015
                             <1>        ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>        ; 20/06/2013 - 24/03/2014 (Retro UNIX 8086 v1)
                             <1>        ;
                             <1>        ; 'sysbreak' sets the programs break points.
                             <1>        ; It checks the current break point (u.break) to see if it is
                             <1>        ; between "core" and the stack (sp). If it is, it is made an
                             <1>        ; even address (if it was odd) and the area between u.break
                             <1>        ; and the stack is cleared. The new breakpoint is then put
                             <1>        ; in u.break and control is passed to 'sysret'.
                             <1>        ;
                             <1>        ; Calling sequence:
                             <1>        ;      sysbreak; addr
                             <1>        ; Arguments: -
                             <1>        ;
                             <1>        ; Inputs: u.break - current breakpoint
                             <1>        ; Outputs: u.break - new breakpoint
                             <1>        ;      area between old u.break and the stack (sp) is cleared.
                             <1>        ; ...............................................................
                             <1>        ;
                             <1>        ; Retro UNIX 8086 v1 modification:
                             <1>        ;      The user/application program puts breakpoint address
                             <1>        ;       in BX register as 'sysbreak' system call argument.
                             <1>        ;      (argument transfer method 1)
                             <1>        ;
                             <1>        ;  NOTE: Beginning of core is 0 in Retro UNIX 8086 v1 !
                             <1>        ;      ((!'sysbreak' is not needed in Retro UNIX 8086 v1!))
                             <1>        ;  NOTE:
                             <1>        ;      'sysbreak' clears extended part (beyond of previous
                             <1>        ;      'u.break' address) of user's memory for original unix's
                             <1>        ;      'bss' compatibility with Retro UNIX 8086 v1 (19/11/2013)
                             <1>
                             <1>               ; mov u.break,r1 / move users break point to r1
                             <1>               ; cmp r1,$core / is it the same or lower than core?
                             <1>               ; blos 1f / yes, 1f
                             <1>        ; 23/06/2015
0000DD7B 8B2D[90030300]      <1>        mov    ebp, [u.break] ; virtual address (offset)
                             <1>        ;and   ebp, ebp
                             <1>        ;jz    short sysbreak_3
                             <1>        ; Retro UNIX 386 v1 NOTE: u.break points to virtual address !!!
                             <1>        ; (Even break point address is not needed for Retro UNIX 386 v1)
0000DD81 8B15[5C030300]      <1>        mov    edx, [u.sp] ; kernel stack at the beginning of sys call
0000DD87 83C20C              <1>        add    edx, 12 ; EIP -4-> CS -4-> EFLAGS -4-> ESP (user)
                             <1>        ; 07/10/2015
0000DD8A 891D[90030300]      <1>        mov    [u.break], ebx ; virtual address !!!
```

```
                             <1>       ;
0000DD90 3B1A                <1>       cmp    ebx, [edx] ; compare new break point with
                             <1>                       ; with top of user's stack (virtual!)
0000DD92 7323                <1>       jnb    short sysbreak_3
                             <1>              ; cmp r1,sp / is it the same or higher
                             <1>                      ; / than the stack?
                             <1>              ; bhis 1f / yes, 1f
0000DD94 89DE                <1>       mov    esi, ebx
0000DD96 29EE                <1>       sub    esi, ebp ; new break point - old break point
0000DD98 761D                <1>       jna    short sysbreak_3
                             <1>       ;push  ebx
                             <1> sysbreak_1:
0000DD9A 89EB                <1>       mov    ebx, ebp
0000DD9C E8EE74FFFF          <1>       call   get_physical_addr ; get physical address
0000DDA1 72C3                <1>       jc     tr_addr_nm_err
                             <1>       ; 18/10/2015
0000DDA3 89C7                <1>       mov    edi, eax
0000DDA5 29C0                <1>       sub    eax, eax ; 0
                             <1>               ; ECX = remain byte count in page (1-4096)
0000DDA7 39CE                <1>       cmp    esi, ecx
0000DDA9 7302                <1>       jnb    short sysbreak_2
0000DDAB 89F1                <1>       mov    ecx, esi
                             <1> sysbreak_2:
0000DDAD 29CE                <1>       sub    esi, ecx
0000DDAF 01CD                <1>       add    ebp, ecx
0000DDB1 F3AA                <1>       rep    stosb
0000DDB3 09F6                <1>       or     esi, esi
0000DDB5 75E3                <1>       jnz    short sysbreak_1
                             <1>       ;
                             <1>              ; bit $1,r1 / is it an odd address
                             <1>              ; beq 2f / no, its even
                             <1>              ; clrb (r1)+ / yes, make it even
                             <1>       ; 2: / clear area between the break point and the stack
                             <1>              ; cmp r1,sp / is it higher or same than the stack
                             <1>              ; bhis 1f / yes, quit
                             <1>              ; clr (r1)+ / clear word
                             <1>              ; br 2b / go back
                             <1>       ;pop   ebx
                             <1> sysbreak_3: ; 1:
                             <1>       ;mov   [u.break], ebx ; virtual address !!!
                             <1>              ; jsr r0,arg; u.break / put the "address"
                             <1>                     ; / in u.break (set new break point)
                             <1>              ; br sysret4 / br sysret
0000DDB7 E922E9FFFF          <1>       jmp    sysret
                             <1>
                             <1> sysseek: ; / moves read write pointer in an fsp entry
                             <1>       ; 06/11/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'sysseek' changes the r/w pointer of (3rd word of in an
                             <1>       ; fsp entry) of an open file whose file descriptor is in u.r0.
                             <1>       ; The file descriptor refers to a file open for reading or
                             <1>       ; writing. The read (or write) pointer is set as follows:
                             <1>       ;      * if 'ptrname' is 0, the pointer is set to offset.
                             <1>       ;      * if 'ptrname' is 1, the pointer is set to its
                             <1>       ;        current location plus offset.
                             <1>       ;      * if 'ptrname' is 2, the pointer is set to the
                             <1>       ;        size of file plus offset.
                             <1>       ; The error bit (e-bit) is set for an undefined descriptor.
                             <1>       ;
                             <1>       ; Calling sequence:
                             <1>       ;      sysseek; offset; ptrname
                             <1>       ; Arguments:
                             <1>       ;      offset - number of bytes desired to move
                             <1>       ;              the r/w pointer
                             <1>       ;      ptrname - a switch indicated above
                             <1>       ;
                             <1>       ; Inputs: r0 - file descriptor
                             <1>       ; Outputs: -
                             <1>       ; ...............................................................
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 modification:
                             <1>       ;       'sysseek' system call has three arguments; so,
                             <1>       ;      * 1st argument, file descriptor is in BX (BL) register
                             <1>       ;      * 2nd argument, offset is in CX register
                             <1>       ;      * 3rd argument, ptrname/switch is in DX (DL) register
                             <1>
0000DDBC E821000000          <1>       call   seektell
                             <1>       ; EAX = Current R/W pointer of the file
                             <1>       ; EBX = [u.fofp]
                             <1>       ; [u.base] = offset (ECX input)
                             <1>
0000DDC1 0305[84030300]      <1>       add    eax, [u.base]
0000DDC7 8903                <1>       mov    [ebx], eax
0000DDC9 E910E9FFFF          <1>       jmp    sysret
                             <1>
                             <1> systell: ; / get the r/w pointer
                             <1>       ; 06/11/2016 - TRDOS 386 (TRDOS v2.0) - temporary !-
                             <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 modification:
                             <1>       ; ! 'systell' does not work in original UNIX v1,
                             <1>       ;          it returns with error !
                             <1>       ; Inputs: r0 - file descriptor
                             <1>       ; Outputs: r0 - file r/w pointer
                             <1>
                             <1>       ;xor   ecx, ecx ; 0
0000DDCE BA01000000          <1>       mov    edx, 1 ; 05/08/2013
                             <1>       ;call  seektell
0000DDD3 E810000000          <1>       call   seektell0 ; 05/08/2013
                             <1>       ;; 06/11/2016
                             <1>       ;; mov eax, [ebx]
0000DDD8 A3[64030300]        <1>       mov    [u.r0], eax
0000DDDD E9FCE8FFFF          <1>       jmp    sysret
                             <1> 
                             <1> ; Original unix v1 'systell' system call:
                             <1>              ; jsr r0,seektell
                             <1>              ; br error4
                             <1> 
                             <1> seektell:
                             <1>       ; 06/11/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 03/01/2016
                             <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'seektell' puts the arguments from sysseek and systell
                             <1>       ; call in u.base and u.count. It then gets the i-number of
                             <1>       ; the file from the file descriptor in u.r0 and by calling
                             <1>       ; getf. The i-node is brought into core and then u.count
                             <1>       ; is checked to see it is a 0, 1, or 2.
                             <1>       ; If it is 0 - u.count stays the same
                             <1>       ;          1 - u.count = offset (u.fofp)
                             <1>       ;         2 - u.count = i.size (size of file)
                             <1>       ;
                             <1>       ; !! Retro UNIX 8086 v1 modification:
                             <1>       ;      Argument 1, file descriptor is in BX;
                             <1>       ;      Argument 2, offset is in CX;
                             <1>       ;      Argument 3, ptrname/switch is in DX register.
                             <1>       ;
                             <1>       ; ((Return -> eax = base for offset (position= base+offset))
                             <1>       ;
0000DDE2 890D[84030300]      <1>       mov    [u.base], ecx ; offset
                             <1> seektell0:
0000DDE8 8915[88030300]      <1>       mov    [u.count], edx
                             <1>       ; EBX = file descriptor (file number)
0000DDEE E8F3FEFFFF          <1>       call   getf1
                             <1>       ; EAX = First cluster of the file
                             <1>       ; EBX = File number (Open file number)
                             <1>       ; [u.fofp] = Pointer to File pointer
                             <1>       ; [i.size] = File size
                             <1> 
0000DDF3 09C0                <1>       or     eax, eax
0000DDF5 7514                <1>       jnz    short seektell1
                             <1> 
0000DDF7 B80A000000          <1>       mov    eax, ERR_FILE_NOT_OPEN
0000DDFC A3[64030300]        <1>       mov    [u.r0], eax
0000DE01 A3[C8030300]        <1>       mov    dword [u.error], eax ; 'file not open !'
0000DE06 E9B3E8FFFF          <1>       jmp    error
                             <1> 
                             <1> seektell1:
0000DE0B 8B1D[74030300]      <1>         mov     ebx, [u.fofp]
0000DE11 803D[88030300]01    <1>       cmp    byte [u.count], 1
0000DE18 7705                <1>       ja     short seektell2
0000DE1A 7409                <1>       je     short seektell3
0000DE1C 31C0                <1>       xor    eax, eax
0000DE1E C3                  <1>       retn
                             <1> 
                             <1> seektell2:
0000DE1F A1[55040300]        <1>         mov        eax, [i.size]
0000DE24 C3                  <1>       retn
                             <1> 
                             <1> seektell3:
0000DE25 8B03                <1>       mov    eax, [ebx]
0000DE27 C3                  <1>       retn
                             <1> 
                             <1> sysintr: ; / set interrupt handling
                             <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 07/07/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'sysintr' sets the interrupt handling value. It puts
                             <1>       ; argument of its call in u.intr then branches into 'sysquit'
                             <1>       ; routine. u.tty is checked if to see if a control tty exists.
                             <1>       ; If one does the interrupt character in the tty buffer is
                             <1>       ; cleared and 'sysret'is called. If one does not exits
                             <1>       ; 'sysret' is just called.
                             <1>       ;
                             <1>       ; Calling sequence:
                             <1>       ;      sysintr; arg
                             <1>       ; Argument:
                             <1>       ;      arg - if 0, interrupts (ASCII DELETE) are ignored.
                             <1>       ;          - if 1, intterupts cause their normal result
                             <1>       ;             i.e force an exit.
                             <1>       ;          - if arg is a location within the program,
                             <1>       ;            control is passed to that location when
                             <1>       ;            an interrupt occurs.
                             <1>       ; Inputs: -
                             <1>       ; Outputs: -
                             <1>       ; ...............................................................
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 modification:
                             <1>       ;       'sysintr' system call sets u.intr to value of BX
                             <1>       ;      then branches into sysquit.
                             <1>       ;
0000DE28 66891D[AA030300]    <1>       mov    [u.intr], bx
                             <1>              ; jsr r0,arg; u.intr / put the argument in u.intr
                             <1>              ; br 1f / go into quit routine
0000DE2F E9AAE8FFFF          <1>       jmp    sysret
                             <1> 
                             <1> sysquit:
                             <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 07/07/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'sysquit' turns off the quit signal. it puts the argument of
                             <1>       ; the call in u.quit. u.tty is checked if to see if a control
```

```
                              <1>       ; tty exists. If one does the interrupt character in the tty
                              <1>       ; buffer is cleared and 'sysret'is called. If one does not exits
                              <1>       ; 'sysret' is just called.
                              <1>       ;
                              <1>       ; Calling sequence:
                              <1>       ;      sysquit; arg
                              <1>       ; Argument:
                              <1>       ;      arg - if 0, this call diables quit signals from the
                              <1>       ;             typewriter (ASCII FS)
                              <1>       ;          - if 1, quits are re-enabled and cause execution to
                              <1>       ;             cease and a core image to be produced.
                              <1>       ;              i.e force an exit.
                              <1>       ;          - if arg is an addres in the program,
                              <1>       ;             a quit causes control to sent to that
                              <1>       ;             location.
                              <1>       ; Inputs: -
                              <1>       ; Outputs: -
                              <1>       ; ...............................................................
                              <1>       ;
                              <1>       ; Retro UNIX 8086 v1 modification:
                              <1>       ;       'sysquit' system call sets u.quit to value of BX
                              <1>       ;      then branches into 'sysret'.
                              <1>       ;
0000DE34 66891D[AC030300]     <1>       mov    [u.quit], bx
0000DE3B E99EE8FFFF           <1>       jmp    sysret
                              <1>              ; jsr r0,arg; u.quit / put argument in u.quit
                              <1>       ;1:
                              <1>              ; mov u.ttyp,r1 / move pointer to control tty buffer
                              <1>                           ; / to r1
                              <1>              ; beq sysret4 / return to user
                              <1>              ; clrb 6(r1) / clear the interrupt character
                              <1>                        ; / in the tty buffer
                              <1>              ; br sysret4 / return to user
                              <1>
                              <1> anyi:
                              <1>       ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
                              <1>       ; Major Modification!
                              <1>       ; TRDOS 386 does not permit to delete a file while it is open
                              <1>       ; The role of 'anyi' procedure has been changed to ensure that.
                              <1>       ;
                              <1>       ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>       ; 25/04/2013 (Retro UNIX 8086 v1)
                              <1>       ;
                              <1>       ; 'anyi' is called if a file deleted while open.
                              <1>       ; "anyi" checks to see if someone else has opened this file.
                              <1>       ;
                              <1>       ; INPUTS ->
                              <1>       ;    r1 - contains an i-number
                              <1>       ;    fsp - start of table containing open files
                              <1>       ;
                              <1>       ; OUTPUTS ->
                              <1>       ;    "deleted" flag set in fsp entry of another occurrence of
                              <1>       ;         this file and r2 points 1st word of this fsp entry.
                              <1>       ;    if file not found - bit in i-node map is cleared
                              <1>       ;                     (i-node is freed)
                              <1>       ;                all blocks related to i-node are freed
                              <1>       ;              all flags in i-node are cleared
                              <1>       ; ((AX = R1)) input
                              <1>       ;
                              <1>       ;    (Retro UNIX Prototype : 02/12/2012, UNIXCOPY.ASM)
                              <1>         ;    ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
                              <1>       ;
                              <1>       ; / r1 contains an i-number
                              <1>
                              <1>       ; TRDOS 386 (06/10/2016)
                              <1>       ;
                              <1>       ; INPUT:
                              <1>       ;      EAX = First Cluster
                              <1>       ;       DL = Logical DOS Drive Number
                              <1>       ;
                              <1>       ; OUTPUT:
                              <1>       ;      CF = 1 -> EBX = File Handle/Number/Index
                              <1>       ;      CF = 0 -> EBX = 0
                              <1>       ;
                              <1>       ; Modified Registers: EBX
                              <1>
0000DE40 31DB                 <1>       xor    ebx, ebx
                              <1> anyi_0:
0000DE42 80BB[62690100]00     <1>       cmp    byte [ebx+OF_MODE], 0 ; 0 = empty entry
0000DE49 770A                 <1>       ja     short anyi_2 ; 1 (r), 2 (w) or 3 (r&w)
                              <1> anyi_1:
0000DE4B FEC3                 <1>       inc    bl
0000DE4D 80FB0A               <1>       cmp    bl, OPENFILES ; max. count of open files
0000DE50 72F0                 <1>       jb     short anyi_0
0000DE52 31C0                 <1>       xor    eax, eax
0000DE54 C3                   <1>       retn
                              <1> anyi_2:
0000DE55 3A93[58690100]       <1>       cmp    dl, [ebx+OF_DRIVE]
0000DE5B 75EE                 <1>       jne    short anyi_1
0000DE5D 66C1E302             <1>       shl    bx, 2 ; *4 (dword offset)
0000DE61 3B83[30690100]       <1>       cmp    eax, [ebx+OF_FCLUSTER]
0000DE67 7406                 <1>       je     short anyi_3
0000DE69 66C1EB02             <1>       shr    bx, 2 ; /4 (byte offset)
0000DE6D EBDC                 <1>       jmp    short anyi_1
                              <1> anyi_3:
0000DE6F 66C1EB02             <1>       shr    bx, 2 ; /4 (bytes offset) (index)
0000DE73 F9                   <1>       stc
0000DE74 C3                   <1>       retn
                              <1>
                              <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS9.INC
                              <1> ; Last Modification: 09/12/2015
                              <1>
                              <1> syssleep:
```

```
                             <1>       ; 29/06/2015 - (Retro UNIX 386 v1)
                             <1>       ; 11/06/2014 - (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 feature only
                             <1>       ; (INPUT -> none)
                             <1>       ;
0000DE75 0FB61D[B3030300]    <1>       movzx  ebx, byte [u.uno] ; process number
0000DE7C 8AA3[7F000300]      <1>       mov    ah, [ebx+p.ttyc-1] ; current/console tty
0000DE82 E841190000          <1>       call   sleep
0000DE87 E952E8FFFF          <1>       jmp    sysret
                             <1>
                             <1> _vp_clr:
                             <1>       ; Reset/Clear Video Page
                             <1>       ;
                             <1>       ; 30/06/2015 - (Retro UNIX 386 v1)
                             <1>       ; 21/05/2013 - 30/10/2013(Retro UNIX 8086 v1) (U0.ASM)
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 feature only !
                             <1>       ;
                             <1>       ; INPUTS ->
                             <1>       ;   BH = video page number
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;   none
                             <1>       ; ((Modified registers: eAX, BH, eCX, eDX, eSI, eDI))
                             <1>       ;
                             <1>       ; 04/12/2013
0000DE8C 28C0                <1>       sub    al, al
                             <1>       ; al = 0 (clear video page)
                             <1>       ; bh = video page ; 13/05/2016
0000DE8E B407                <1>       mov    ah, 07h
                             <1>       ; ah = 7 (attribute/color)
0000DE90 6631C9              <1>       xor    cx, cx ; 0, left upper column (cl) & row (cl)
0000DE93 66BA4F18            <1>       mov    dx, 184Fh ; right lower column & row (dl=24, dh=79)
0000DE97 E86E3BFFFF          <1>       call   _scroll_up
                             <1>       ; bh = video page
0000DE9C 6631D2              <1>       xor    dx, dx ; 0 (cursor position)
0000DE9F E9A43EFFFF          <1>       jmp    _set_cpos
                             <1>
                             <1> sysmsg:
                             <1>       ; 13/05/2016
                             <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 01/07/2015 - 11/11/2015 (Retro UNIX 386 v1)
                             <1>       ; Print user-application message on user's console tty
                             <1>       ;
                             <1>       ; Input -> EBX = Message address
                             <1>       ;         ECX = Message length (max. 255)
                             <1>       ;         DL = Color (IBM PC Rombios color attributes)
                             <1>       ;
0000DEA4 81F9FF000000        <1>       cmp    ecx, MAX_MSG_LEN ; 255
0000DEAA 0F872EE8FFFF        <1>       ja     sysret ; nothing to do with big message size
0000DEB0 08C9                <1>       or     cl, cl
0000DEB2 0F8426E8FFFF        <1>       jz     sysret
0000DEB8 20D2                <1>       and    dl, dl
0000DEBA 7502                <1>       jnz    short sysmsg0
0000DEBC B207                <1>       mov    dl, 07h ; default color
                             <1>              ; (black background, light gray character)
                             <1> sysmsg0:
0000DEBE 891D[84030300]      <1>       mov    [u.base], ebx
0000DEC4 8815[67580100]      <1>       mov    [ccolor], dl ; color attributes
0000DECA 89E5                <1>       mov    ebp, esp
0000DECC 31DB                <1>       xor    ebx, ebx ; 0
0000DECE 891D[8C030300]      <1>       mov    [u.nread], ebx ; 0
                             <1>       ;
0000DED4 381D[C6030300]      <1>       cmp    [u.kcall], bl ; 0
0000DEDA 7769                <1>       ja     short sysmsgk ; Temporary (01/07/2015)
                             <1>       ;
0000DEDC 890D[88030300]      <1>       mov    [u.count], ecx
0000DEE2 41                  <1>       inc    ecx ; + 00h ; ASCIIZ
0000DEE3 29CC                <1>       sub    esp, ecx
0000DEE5 89E7                <1>       mov    edi, esp
0000DEE7 89E6                <1>       mov    esi, esp
0000DEE9 66891D[C4030300]    <1>       mov    [u.pcount], bx ; reset page (phy. addr.) counter
                             <1>       ; 11/11/2015
0000DEF0 8A25[94030300]      <1>       mov    ah, [u.ttyp] ; recent open tty
                             <1>       ; 0 = none
0000DEF6 FECC                <1>       dec    ah
0000DEF8 790C                <1>       jns    short sysmsg1
0000DEFA 8A1D[B3030300]      <1>       mov    bl, [u.uno] ; process number
0000DF00 8AA3[7F000300]      <1>       mov    ah, [ebx+p.ttyc-1] ; user's (process's) console tty
                             <1> sysmsg1:
0000DF06 8825[96030300]      <1>       mov    [u.ttyn], ah
                             <1> sysmsg2:
0000DF0C E808080000          <1>       call   cpass
0000DF11 7416                <1>       jz     short sysmsg5
0000DF13 AA                  <1>       stosb
0000DF14 20C0                <1>       and    al, al
0000DF16 75F4                <1>       jnz    short sysmsg2
                             <1> sysmsg3:
0000DF18 80FC07              <1>       cmp    ah, 7 ; tty number
0000DF1B 7711                <1>       ja     short sysmsg6 ; serial port
0000DF1D E83E000000          <1>       call   print_cmsg
                             <1> sysmsg4:
0000DF22 89EC                <1>       mov    esp, ebp
0000DF24 E9B5E7FFFF          <1>       jmp    sysret
                             <1> sysmsg5:
0000DF29 C60700              <1>       mov    byte [edi], 0
0000DF2C EBEA                <1>       jmp    short sysmsg3
                             <1> sysmsg6:
0000DF2E 8A06                <1>       mov    al, [esi]
0000DF30 E891180000          <1>       call   sndc
0000DF35 72EB                <1>       jc     short sysmsg4
0000DF37 803E00              <1>       cmp    byte [esi], 0  ; 0 is stop character
```

```
0000DF3A 76E6                <1>       jna    short sysmsg4
0000DF3C 46                  <1>       inc    esi
0000DF3D 8A25[96030300]      <1>       mov    ah, [u.ttyn]
0000DF43 EBE9                <1>       jmp    short sysmsg6
                             <1> 
                             <1> sysmsgk: ; Temporary (01/07/2015)
                             <1>       ; The message has been sent by Kernel (ASCIIZ string)
                             <1>       ; (ECX -character count- will not be considered)
0000DF45 8B35[84030300]      <1>       mov    esi, [u.base]
0000DF4B 8A25[66580100]      <1>       mov    ah, [ptty] ; present/current screen (video page)
0000DF51 8825[96030300]      <1>       mov    [u.ttyn], ah
0000DF57 C605[C6030300]00    <1>       mov    byte [u.kcall], 0
0000DF5E EBB8                <1>       jmp    short sysmsg3
                             <1> 
                             <1> print_cmsg:
                             <1>       ; 18/11/2017
                             <1>       ; 13/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 01/07/2015 (Retro UNIX 386 v1)
                             <1>       ;
                             <1>       ; print message (on user's console tty)
                             <1>       ;      with requested color
                             <1>       ;
                             <1>       ; INPUTS:
                             <1>       ;      esi = message address
                             <1>       ;      [u.ttyn] = tty number (0 to 7)
                             <1>       ;      [ccolor] = color attributes (IBM PC BIOS colors)
                             <1> 
                             <1>       ;mov   bh, ah
0000DF60 8A3D[96030300]      <1>       mov    bh, [u.ttyn]
                             <1>       ;mov   bl, [ccolor] ; *
                             <1> pcmsg1:
0000DF66 AC                  <1>       lodsb
0000DF67 20C0                <1>       and    al, al  ; 0
0000DF69 740F                <1>       jz     short pcmsg2
0000DF6B 56                  <1>       push   esi
0000DF6C 8A1D[67580100]      <1>       mov    bl, [ccolor]  ; * (video.s 'u11'&'beep' change BL)
                             <1>       ;mov   bh, [u.ttyn]
0000DF72 E83B3DFFFF          <1>       call   _write_tty
0000DF77 5E                  <1>       pop    esi
0000DF78 EBEC                <1>       jmp    short pcmsg1
                             <1> pcmsg2:
0000DF7A C3                  <1>       retn
                             <1> 
                             <1> sysgeterr:
                             <1>       ; 09/12/2015
                             <1>       ; 21/09/2015 - (Retro UNIX 386 v1 feature only!)
                             <1>       ; Get last error number or page fault count
                             <1>       ; (for debugging)
                             <1>       ;
                             <1>       ; Input -> EBX = return type
                             <1>       ;         0 = last error code (which is in 'u.error')
                             <1>       ;         FFFFFFFFh = page fault count for running process
                             <1>       ;         FFFFFFFEh = total page fault count
                             <1>       ;         1 .. FFFFFFFDh = undefined
                             <1>       ;
                             <1>       ; Output -> EAX = last error number or page fault count
                             <1>       ;         (depending on EBX input)
                             <1>       ;
0000DF7B 21DB                <1>       and    ebx, ebx
0000DF7D 750B                <1>       jnz    short glerr_2
                             <1> glerr_0:
0000DF7F A1[C8030300]        <1>       mov    eax, [u.error]
                             <1> glerr_1:
0000DF84 A3[64030300]        <1>       mov    [u.r0], eax
0000DF89 C3                  <1>       retn
                             <1> glerr_2:
0000DF8A 43                  <1>       inc    ebx ; FFFFFFFFh -> 0, FFFFFFFEh -> FFFFFFFFh
0000DF8B 74FD                <1>       jz     short glerr_2 ; page fault count for process
0000DF8D 43                  <1>       inc    ebx ; FFFFFFFFh -> 0
0000DF8E 75EF                <1>       jnz    short glerr_0
0000DF90 A1[80050300]        <1>       mov    eax, [PF_Count] ; total page fault count
0000DF95 EBED                <1>         jmp     short glerr_1
                             <1> glerr_3:
0000DF97 A1[CC030300]        <1>       mov    eax, [u.pfcount]
0000DF9C EBE6                <1>       jmp    short glerr_1
                             <1> 
                             <1> load_and_run_file:
                             <1>       ; 18/11/2017
                             <1>       ; 22/01/2017
                             <1>       ; 04/01/2017, 07/01/2017
                             <1>       ; 24/10/2016
                             <1>       ; 24/04/2016, 02/05/2016, 03/05/2016, 06/05/2016
                             <1>       ; 23/04/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 23/10/2015 (Retro UNIX 386 v1, 'sysexec')
                             <1>       ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)
                             <1>       ; EAX = First Cluster number
                             <1>       ; EDX = File Size
                             <1>       ; ESI = Argument list address
                             <1>       ; [argc] = argument count
                             <1>       ; [u.nread] = argument list length
                             <1>       ; [esp] = return address to the caller (*)
                             <1>       ;
0000DF9E 8935[4C040300]      <1>       mov    [argv], esi
0000DFA4 8915[55040300]      <1>       mov    [i.size], edx
0000DFAA A3[51040300]        <1>       mov    [ii], eax
                             <1> 
                             <1>       ;sti   ; 07/01/2017
                             <1>       ;mov   eax, [k_page_dir]
                             <1>       ;mov   [u.pgdir], eax
0000DFAF 31C0                <1>       xor    eax, eax ; clc ; *** ; 04/01/2017
                             <1>       ;mov   [u.r0], eax ; 0 ; 07/01/2017
                             <1> 
                             <1>       ; 06/05/2016
                             <1>       ; Set 'sysexit' return order to MainProg
                             <1>       ;
0000DFB1 58                  <1>       pop    eax ; * 'loc_load_and_run_file_8:' address
                             <1>       ;; 22/01/2017
                             <1>       ;;cli ; 07/01/2017
0000DFB2 8B25[D4570100]      <1>       mov    esp, [tss.esp0]
                             <1>       ;
                             <1>       ; 'loc_load_run_file_8' address has
                             <1>       ; 'jmp loc_file_rw_restore_retn' instruction
                             <1>       ; 'loc_file_rw_restore_retn:' will return to
                             <1>       ; [mainprog_return_addr]
                             <1>       ; just after 'call command_interpreter'
                             <1>       ;
0000DFB8 68[3B630000]        <1>       push   _end_of_mainprog ; we must not return to here !
0000DFBD FF35[BC650100]      <1>       push   dword [mainprog_return_addr]
0000DFC3 89E5                <1>       mov    ebp, esp ; **
                             <1>       ;
0000DFC5 9C                  <1>       pushfd  ; EFLAGS      ; IRETD ; ***
0000DFC6 6A08                <1>       push   KCODE ; cs    ; IRETD
0000DFC8 50                  <1>       push   eax ; * (eip) ; IRETD
0000DFC9 8925[5C030300]      <1>       mov    [u.sp], esp
                             <1>       ;mov   byte [u.quant], time_count
0000DFCF 1E                  <1>       push   ds
0000DFD0 06                  <1>       push   es
0000DFD1 0FA0                <1>       push   fs
0000DFD3 0FA8                <1>       push   gs
                             <1>       ;mov   eax, [u.r0]
0000DFD5 29C0                <1>       sub    eax, eax
0000DFD7 60                  <1>       pushad
0000DFD8 68[DEC60000]        <1>       push   sysret
                             <1>       ;push  sysrel1 ; 07/01/2017
0000DFDD 8925[60030300]      <1>       mov    [u.usp], esp
                             <1>       ;
0000DFE3 E845060000          <1>       call   wswap ; Save MainProg (process 1) 'u' structure
                             <1>                    ; and registers for return (from program)
0000DFE8 89EC                <1>       mov    esp, ebp ; **
                             <1>       ;;22/01/2017
                             <1>       ;;sti ; 07/01/2017
0000DFEA 50                  <1>       push   eax  ; * 'loc_load_and_run_file_8:' address
                             <1>       ;
                             <1>       ;;; 02/05/2016
                             <1>       ;;; Create a new process (parent: MainProg)
0000DFEB 31F6                <1>       xor    esi, esi
                             <1> cnpm_1: ; search p.stat table for unused process number
0000DFED 46                  <1>       inc    esi
0000DFEE 80BE[AF000300]00    <1>       cmp    byte [esi+p.stat-1], 0 ; SFREE
                             <1>                           ; is process active, unused, dead
0000DFF5 760B                <1>       jna    short cnpm_2 ; it's unused so branch
0000DFF7 6683FE10            <1>       cmp    si, nproc    ; all processes checked
0000DFFB 72F0                <1>       jb     short cnpm_1    ; no, branch back
0000DFFD E9AF83FFFF          <1>       jmp    panic
                             <1> cnpm_2:
0000E002 A1[B8030300]        <1>       mov    eax, [u.pgdir] ; page directory of MainProg
0000E007 A3[BC030300]        <1>       mov    [u.ppgdir], eax ; parent's page directory
0000E00C E8696BFFFF          <1>       call   allocate_page
0000E011 0F829A83FFFF        <1>       jc     panic
                             <1>       ; EAX = UPAGE (user structure page) address
0000E017 A3[B4030300]        <1>       mov    [u.upage], eax ; memory page for 'user' struct (child)
0000E01C 89F7                <1>       mov    edi, esi
0000E01E 66C1E702            <1>       shl    di, 2
0000E022 8987[BC000300]      <1>       mov    [edi+p.upage-4], eax ; memory page for 'user' struct
0000E028 E8C76BFFFF          <1>       call   clear_page ; 03/05/2016
                             <1>       ;movzx eax, byte [p.ttyc] ; console tty (for MainProg)
0000E02D 6629C0              <1>       sub    ax, ax ; 0
0000E030 668986[7F000300]    <1>       mov     [esi+p.ttyc-1], ax ; al - set child's console tty
                             <1>                              ; ah - reset child's wait channel
0000E037 89F0                <1>       mov    eax, esi
0000E039 A2[B3030300]        <1>       mov    [u.uno], al ; child process number
0000E03E FE86[AF000300]      <1>         inc     byte [esi+p.stat-1] ; 1, SRUN
0000E044 66D1E6              <1>       shl    si, 1 ; multiply si by 2 to get index into p.pid table
0000E047 66FF05[4E030300]    <1>       inc    word [mpid] ; increment m.pid; get a new process name
0000E04E 66A1[4E030300]      <1>       mov    ax, [mpid]
0000E054 668986[1E000300]    <1>       mov    [esi+p.pid-2], ax ; put new process name
                             <1>                             ; in child process' name slot
                             <1>       ;mov   ax, [p.pid]  ; get process name of MainProg
0000E05B 66B80100            <1>       mov    ax, 1
0000E05F 668986[3E000300]    <1>       mov    [esi+p.ppid-2], ax ; put parent process name
                             <1>                               ; in parent process slot for child
0000E066 6648                <1>       dec    ax ; 0
0000E068 66A3[94030300]      <1>       mov    [u.ttyp], ax ; 0
                             <1>       ;;;
0000E06E A1[51040300]        <1>       mov    eax,  [ii]
                             <1>       ; Retro UNIX 386 v1, 'sysexec' (u2.s)
0000E073 E84C170000          <1>       call   iopen
                             <1>       ; 06/06/2016
0000E078 C605[A9030300]01    <1>       mov    byte [u.pri], 1 ; normal priority
                             <1>       ;
0000E07F EB16                <1>       jmp    short sysexec_7 ; 02/05/2016
                             <1>
                             <1> sysexec_6:
                             <1>       ; 19/11/2017
                             <1>       ; 18/11/2017
                             <1>       ; 14/11/2017
                             <1>       ; 13/11/2017
0000E081 8925[4C040300]      <1>       mov    [argv], esp ; *!* ; start address of argument list
                             <1>
                             <1>       ; 04/01/2017
                             <1>       ; 24/10/2016
                             <1>       ;;02/05/2016
                             <1>       ; 23/04/2016 (TRDOS 386)
                             <1>       ; 18/10/2015 ('sysexec_6')
                             <1>       ; 23/06/2015
```

```
0000E087 A1[B8030300]        <1>       mov    eax, [u.pgdir] ; physical address of page directory
                             <1>       ;cmp   eax, [k_page_dir] ; TRDOS MainProg ?
                             <1>       ;je    short sysexec_7
                             <1>       ; 19/11/2017
0000E08C 8B1D[BC030300]      <1>       mov    ebx, [u.ppgdir] ; phy addr of the parent's page dir
0000E092 E81C6CFFFF          <1>       call   deallocate_page_dir
                             <1> sysexec_7:
0000E097 E84C6BFFFF          <1>       call   make_page_dir
0000E09C 0F820F83FFFF        <1>       jc     panic  ; allocation error
                             <1>                     ; after a deallocation would be nonsence !?
                             <1>       ; 24/07/2015
                             <1>       ; map kernel pages (1st 4MB) to PDE 0
                             <1>       ;     of the user's page directory
                             <1>       ;     (It is needed for interrupts!)
                             <1>       ; 18/10/2015
0000E0A2 8B15[38580100]      <1>       mov    edx, [k_page_dir] ; Kernel's page directory
0000E0A8 8B02                <1>       mov    eax, [edx] ; physical address of
                             <1>                       ; kernel's first page table (1st 4 MB)
                             <1>                       ; (PDE 0 of kernel's page directory)
0000E0AA 8B15[B8030300]      <1>       mov    edx, [u.pgdir]
0000E0B0 8902                <1>       mov    [edx], eax ; PDE 0 (1st 4MB)
                             <1>       ;
                             <1>       ; 20/07/2015
0000E0B2 BB00004000          <1>       mov    ebx, CORE ; start address = 0 (virtual) + CORE
                             <1>       ; 18/10/2015
0000E0B7 BE[3C040300]        <1>       mov    esi, pcore ; physical start address
                             <1> sysexec_8:
0000E0BC B907000000          <1>       mov    ecx, PDE_A_USER + PDE_A_WRITE + PDE_A_PRESENT
0000E0C1 E8406BFFFF          <1>       call   make_page_table
0000E0C6 0F82E582FFFF        <1>       jc     panic
                             <1>       ;mov   ecx, PTE_A_USER + PTE_A_WRITE + PTE_A_PRESENT
0000E0CC E8436BFFFF          <1>       call   make_page ; make new page, clear and set the pte
0000E0D1 0F82DA82FFFF        <1>       jc     panic
                             <1>       ;
0000E0D7 8906                <1>       mov    [esi], eax ; 24/06/2015
                             <1>       ; ebx = virtual address (24/07/2015)
0000E0D9 E8DB70FFFF          <1>       call   add_to_swap_queue
                             <1>       ; 18/10/2015
0000E0DE 81FE[40040300]      <1>       cmp    esi, ecore ; user's stack (last) page ?
0000E0E4 740C                <1>       je     short sysexec_9 ; yes
0000E0E6 BE[40040300]        <1>       mov    esi, ecore  ; physical address of the last page
                             <1>       ; 20/07/2015
0000E0EB BB00F0FFFF          <1>       mov    ebx, (ECORE - PAGE_SIZE) + CORE
                             <1>       ; ebx = virtual end address + segment base address - 4K
0000E0F0 EBCA                <1>         jmp     short sysexec_8
                             <1> sysexec_9:
                             <1>       ; 19/11/2017
                             <1>       ; 24/04/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; 25/06/2015, 26/08/2015, 18/10/2015
                             <1>       ; move arguments from kernel stack to [ecore]
                             <1>       ; (argument list/line will be copied from kernel stack
                             <1>       ; frame to the last (stack) page of user's core memory)
                             <1>       ; 18/10/2015
0000E0F2 8B3D[40040300]      <1>       mov    edi, [ecore]
0000E0F8 81C700100000        <1>       add    edi, PAGE_SIZE
                             <1>       ; 19/11/2017
0000E0FE 83EF04              <1>       sub    edi, 4
0000E101 C70700000000        <1>       mov    dword [edi], 0
0000E107 89FB                <1>       mov    ebx, edi
                             <1>       ;
0000E109 0FB705[4A040300]    <1>       movzx  eax, word [argc]
0000E110 09C0                <1>       or     eax, eax
0000E112 7445                <1>       jz     short sysexec_13 ; 19/11/2017
                             <1>       ;jnz   short sysexec_10
                             <1>       ;mov   ebx, edi
                             <1>       ;sub   ebx, 4
                             <1>       ;mov   [ebx], eax ; 0
                             <1>       ;jmp   short sysexec_13
                             <1> sysexec_10:
0000E114 8B0D[8C030300]      <1>       mov    ecx, [u.nread]
                             <1>       ; 13/11/2017
                             <1>       ;mov   esi, TextBuffer ; 'load_and_execute_file'
                             <1>       ;mov   esi, esp     ; 'sysexec'
0000E11A 8B35[4C040300]      <1>       mov    esi, [argv] ; 24/04/2016 (TRDOS 386  = TRDOS v2.0)
                             <1>       ;sub   edi, ecx ; page end address - argument list length
0000E120 29CB                <1>       sub    ebx, ecx ; 19/11/2017
0000E122 89C2                <1>       mov    edx, eax
0000E124 FEC2                <1>       inc    dl ; argument count + 1 for argc value
0000E126 C0E202              <1>       shl    dl, 2  ; 4 * (argument count + 1)
                             <1>       ;mov   ebx, edi
0000E129 89DF                <1>       mov    edi, ebx ; 19//11/2017
0000E12B 80E3FC              <1>       and    bl, 0FCh ; 32 bit (dword) alignment
0000E12E 29D3                <1>       sub    ebx, edx
0000E130 89FA                <1>       mov    edx, edi
0000E132 F3A4                <1>       rep    movsb
0000E134 89D6                <1>       mov    esi, edx
0000E136 89DF                <1>       mov    edi, ebx
0000E138 BA00F0BFFF          <1>       mov    edx, ECORE - PAGE_SIZE ; virtual addr. of the last page
0000E13D 2B15[40040300]      <1>       sub    edx, [ecore] ; difference (virtual - physical)
0000E143 AB                  <1>       stosd  ; eax = argument count
                             <1> sysexec_11:
0000E144 89F0                <1>       mov    eax, esi
0000E146 01D0                <1>       add    eax, edx
0000E148 AB                  <1>       stosd  ; eax = virtual address
                             <1>       ;dec   byte [argc]
0000E149 66FF0D[4A040300]    <1>       dec    word [argc] ; 14/11/2017
0000E150 7407                <1>       jz     short sysexec_13
                             <1> sysexec_12:
0000E152 AC                  <1>       lodsb
0000E153 20C0                <1>       and    al, al
0000E155 75FB                <1>       jnz    short sysexec_12
0000E157 EBEB                <1>       jmp    short sysexec_11
                             <1> sysexec_13:
```

```
                             <1>        ; 24/10/2016
                             <1>        ; 24/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>        ; 23/06/2015 - 19/10/2015 (Retro UNIX 386 v1, 'sysexec_13')
                             <1>        ;
                             <1>        ; moving arguments to [ecore] is OK here..
                             <1>        ;
                             <1>        ; ebx = beginning addres of argument list pointers
                             <1>                ;      in user's stack
0000E159 2B1D[40040300]      <1>        sub    ebx, [ecore]
0000E15F 81C300F0BFFF        <1>        add      ebx, (ECORE - PAGE_SIZE)
                             <1>                        ; end of core - 4096 (last page)
                             <1>                        ; (virtual address)
0000E165 891D[4C040300]      <1>        mov    [argv], ebx
0000E16B 891D[90030300]      <1>        mov    [u.break], ebx ; available user memory
                             <1>        ;
0000E171 29C0                <1>        sub    eax, eax
0000E173 C705[88030300]2000- <1>        mov    dword [u.count], 32 ; Executable file header size
0000E17B 0000                <1>
0000E17D C705[74030300]-     <1>        mov    dword [u.fofp], u.off
0000E183 [80030300]          <1>
0000E187 A3[80030300]        <1>        mov    [u.off], eax ; 0
0000E18C A3[84030300]        <1>        mov    [u.base], eax ; 0, start of user's core (virtual)
                             <1>        ; 24/10/2016
0000E191 A0[FE580100]        <1>        mov    al, [Current_Drv]
0000E196 A2[46030300]        <1>        mov    [cdev], al
                             <1>        ;
0000E19B A1[51040300]        <1>        mov    eax, [ii] ; Fist Cluster of the Program (PRG) file
                             <1>        ; EAX = First cluster of the executable file
0000E1A0 E80A010000          <1>        call   readi
                             <1>
0000E1A5 8B0D[90030300]      <1>        mov    ecx, [u.break] ; top of user's stack (physical addr.)
0000E1AB 890D[88030300]      <1>        mov    [u.count], ecx ; save for overrun check
                             <1>        ;
0000E1B1 8B0D[8C030300]      <1>        mov    ecx, [u.nread]
0000E1B7 890D[90030300]      <1>        mov    [u.break], ecx ; virtual address (offset from start)
0000E1BD 80F920              <1>        cmp    cl, 32
0000E1C0 7540                <1>          jne     short sysexec_15
                             <1>        ;:
                             <1>        ; Retro UNIX 386 v1 (32 bit) executable file header format
0000E1C2 8B35[3C040300]      <1>        mov    esi, [pcore] ; start address of user's core memory
                             <1>                       ; (phys. start addr. of the exec. file)
0000E1C8 AD                  <1>        lodsd
0000E1C9 663DEB1E            <1>        cmp    ax, 1EEBh ; EBH, 1Eh -> jump to +32
0000E1CD 7533                <1>        jne    short sysexec_15
0000E1CF AD                  <1>        lodsd
0000E1D0 89C1                <1>        mov    ecx, eax ; text (code) section size
0000E1D2 AD                  <1>        lodsd
0000E1D3 01C1                <1>        add    ecx, eax ; + data section size (initialized data)
0000E1D5 89CB                <1>        mov    ebx, ecx
0000E1D7 AD                  <1>        lodsd
0000E1D8 01C3                <1>        add    ebx, eax ; + bss section size (for overrun checking)
0000E1DA 3B1D[88030300]      <1>        cmp    ebx, [u.count]
0000E1E0 7711                <1>        ja     short sysexec_14  ; program overruns stack !
                             <1>        ;
                             <1>        ; add bss section size to [u.break]
0000E1E2 0105[90030300]      <1>        add    [u.break], eax
                             <1>        ;
0000E1E8 83E920              <1>        sub    ecx, 32  ; header size (already loaded)
                             <1>        ;cmp   ecx, [u.count]
                             <1>        ;jnb   short sysexec_16
0000E1EB 890D[88030300]      <1>        mov    [u.count], ecx ; required read count
0000E1F1 EB29                <1>        jmp    short sysexec_16
                             <1> sysexec_14:
                             <1>        ; insufficient (out of) memory
0000E1F3 C705[C8030300]0400- <1>        mov    dword [u.error], ERR_MINOR_IM ; 1
0000E1FB 0000                <1>
0000E1FD E9BCE4FFFF          <1>        jmp    error
                             <1> sysexec_15:
0000E202 8B15[55040300]      <1>          mov  edx, [i.size] ; file size
0000E208 29CA                <1>        sub    edx, ecx ; file size - loaded bytes
0000E20A 7626                <1>        jna    short sysexec_17 ; no need to next read
0000E20C 01D1                <1>        add    ecx, edx ; [i.size]
0000E20E 3B0D[88030300]      <1>        cmp    ecx, [u.count] ; overrun check (!)
0000E214 77DD                <1>        ja     short sysexec_14
0000E216 8915[88030300]      <1>        mov    [u.count], edx
                             <1> sysexec_16:
0000E21C A1[51040300]        <1>        mov    eax, [ii] ; first cluster
0000E221 E889000000          <1>        call   readi
0000E226 8B0D[8C030300]      <1>        mov    ecx, [u.nread]
0000E22C 010D[90030300]      <1>        add    [u.break], ecx
                             <1> sysexec_17:
0000E232 A1[51040300]        <1>        mov    eax, [ii] ; first cluster
0000E237 E889150000          <1>        call   iclose
0000E23C 31C0                <1>        xor      eax, eax
0000E23E FEC0                <1>        inc    al
0000E240 66A3[AA030300]      <1>        mov    [u.intr], ax ; 1 (interrupt/time-out is enabled)
0000E246 66A3[AC030300]      <1>        mov    [u.quit], ax ; 1 ('crtl+brk' signal is enabled)
0000E24C 833D[BC030300]00    <1>          cmp  dword [u.ppgdir], 0  ; is the caller MainProg (kernel) ?
0000E253 770C                <1>        ja     short sysexec_18 ; no, the caller is user process
                             <1>        ; If the caller is kernel (MainProg), 'sysexec' will come here
0000E255 8B15[38580100]      <1>        mov    edx, [k_page_dir] ; kernel's page directory
0000E25B 8915[BC030300]      <1>        mov    [u.ppgdir], edx ; next time 'sysexec' must not come here
                             <1> sysexec_18:
                             <1>        ; 02/05/2016
                             <1>        ; 24/04/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>        ; 18/10/2015 (Retro UNIX 386 v1)
                             <1>        ; 05/08/2015
                             <1>        ; 29/07/2015
                             <1>
                             <1> ;      ; **** arguments list test start - 19/11/2017
                             <1> ;      mov    ebp, [argv]
                             <1> ;      sub    ebp, ECORE - 4096
                             <1> ;      add    ebp, [ecore]
                             <1> ;
                             <1> ;     mov    ebx, [ebp]
                             <1> ;     mov    [argc], bx
                             <1> ;     add    ebp, 4
                             <1> ;     mov    byte [ccolor], 1Fh
                             <1> ;_zx0:
                             <1> ;     cmp    word [argc], 0
                             <1> ;     jna    short _zx2
                             <1> ;_zx1:
                             <1> ;     push   ebp
                             <1> ;     mov    esi, [ebp]
                             <1> ;
                             <1> ;     sub    esi, ECORE - 4096
                             <1> ;     add    esi, [ecore]
                             <1> ;
                             <1> ;     call   print_cmsg
                             <1> ;
                             <1> ;     dec    word [argc]
                             <1> ;     jz     short _zx2
                             <1> ;
                             <1> ;     mov    al, '.'
                             <1> ;     mov    bl, 07h
                             <1> ;     mov    bh, [u.ttyn]
                             <1> ;     call   _write_tty
                             <1> ;
                             <1> ;     pop    ebp
                             <1> ;     add    ebp, 4
                             <1> ;     jmp    short _zx1
                             <1> ;_zx2:
                             <1> ;     pop    ebp
                             <1> ;     mov    byte [ccolor], 07h
                             <1> ;     mov    eax, 1
                             <1> ;     ; **** arguments list test stop
                             <1> ;     Test result is OK! (there is not a wrong thing) - 19/11/2017
                             <1>
0000E261 8B2D[4C040300]      <1>       mov    ebp, [argv] ; user's stack pointer must point to argument
                             <1>                         ; list pointers (argument count)
0000E267 FA                  <1>       cli
0000E268 8B25[D4570100]      <1>         mov     esp, [tss.esp0] ; ring 0 (kernel) stack pointer
                             <1>       ;mov         esp, [u.sp] ; Restore Kernel stack
                             <1>                         ; for this process
                             <1>       ;add   esp, 20 ; --> EIP, CS, EFLAGS, ESP, SS
                             <1>       ;xor   eax, eax ; 0
0000E26E FEC8                <1>       dec    al ; eax = 0
                             <1>       ;mov   edx, UDATA
                             <1>       ; 18/11/2017
0000E270 6A23                <1>       push   UDATA ; user's stack segment
                             <1>       ;push  edx
0000E272 55                  <1>       push   ebp ; user's stack pointer
                             <1>                  ; (points to number of arguments)
                             <1>
                             <1>       ; 04/01/2017
                             <1>       ; MainProg comes here while [sysflg]= 0FFh
                             <1>       ; (but sysexec comes here while [sysflg]= 0)
0000E273 C605[5B030300]00    <1>       mov    byte [sysflg], 0 ; 04/01/2017
                             <1>                            ; (timer_int sysflg control)
0000E27A FB                  <1>       sti
0000E27B 9C                  <1>       pushfd ; EFLAGS
                             <1>             ; Set IF for enabling interrupts in user mode
                             <1>       ;or    dword [esp], 200h
                             <1>       ;
                             <1>       ;mov   bx, UCODE
                             <1>       ;push  bx ; user's code segment
0000E27C 6A1B                <1>       push   UCODE
                             <1>       ;push  0
0000E27E 50                  <1>       push   eax ; EIP (=0) - start address -
0000E27F 8925[5C030300]      <1>       mov    [u.sp], esp ; 29/07/2015
                             <1>       ; 05/08/2015
                             <1>       ; Remedy of a General Protection Fault during 'iretd' is here !
                             <1>       ; ('push dx' would cause to general protection fault,
                             <1>       ; after 'pop ds' etc.)
                             <1>       ;
                             <1>       ;; push dx ; ds (UDATA)
                             <1>       ;; push dx ; es (UDATA)
                             <1>       ;; push dx ; fs (UDATA)
                             <1>       ;; push dx ; gs (UDATA)
                             <1>       ;
                             <1>       ; This is a trick to prevent general protection fault
                             <1>       ; during 'iretd' intruction at the end of 'sysrele' (in u1.s):
0000E285 66BA2300            <1>       mov    dx, UDATA ; 19/11/2017
0000E289 8EC2                <1>       mov    es, dx ; UDATA
0000E28B 06                  <1>       push   es ; ds (UDATA)
0000E28C 06                  <1>       push   es ; es (UDATA)
0000E28D 06                  <1>       push   es ; fs (UDATA)
0000E28E 06                  <1>       push   es ; gs (UDATA)
0000E28F 66BA1000            <1>       mov    dx, KDATA
0000E293 8EC2                <1>       mov    es, dx
                             <1>       ;
                             <1>       ;; pushad simulation
0000E295 89E5                <1>       mov    ebp, esp ; esp before pushad
0000E297 50                  <1>       push   eax ; eax (0)
0000E298 50                  <1>       push   eax ; ecx (0)
0000E299 50                  <1>       push   eax ; edx (0)
0000E29A 50                  <1>       push   eax ; ebx (0)
0000E29B 55                  <1>       push   ebp ; esp before pushad
0000E29C 50                  <1>       push   eax ; ebp (0)
0000E29D 50                  <1>       push   eax ; esi (0)
0000E29E 50                  <1>       push   eax ; edi (0)
                             <1>       ;
0000E29F A3[64030300]        <1>       mov    [u.r0], eax ; eax = 0
0000E2A4 8925[60030300]      <1>       mov    [u.usp], esp
                             <1>
                             <1>       ; 14/11/2017
```

```
0000E2AA E931E4FFFF          <1>       jmp    sysret0
                             <1> 
                             <1> ;     ; 02/05/2016
                             <1> ;     ;inc   byte [sysflg] ; 0FFh -> 0
                             <1> ;     ;mov   byte [sysflg], 0 ; 04/01/2017
                             <1> ;     movzx  ebx, byte [u.uno]
                             <1> ;     shl    bl, 1 ; 13/11/2017
                             <1> ;     cmp    word [ebx+p.ppid-2], 1 ; MainProg
                             <1> ;     ja     sysret0 ; 03/05/2016
                             <1> ;     push   sysret ; *
                             <1> ;     mov    [u.usp], esp
                             <1> ;     call   wswap ; save child process 'u' structure and
                             <1> ;                  ; registers
                             <1> ;     add    dword [u.usp], 4 ; 03/05/2016
                             <1> ;sysexec_19: ; 02/05/2016
                             <1> ;     retn ; * 'sysret' ; byte [sysflg] -> 0FFh
                             <1> 
                             <1> readi:
                             <1>       ; 01/05/2016
                             <1>       ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 20/05/2015 - Retro UNIX 386 v1
                             <1>       ; 11/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; Reads from a file whose the first cluster number in EAX
                             <1>       ;
                             <1>       ; INPUTS ->
                             <1>       ;    EAX - First cluster number of the file
                             <1>       ;    u.count - byte count user desires
                             <1>       ;    u.base - points to user buffer
                             <1>       ;    u.fofp - points to dword with current file offset
                             <1>       ;    i.size - file size
                             <1>       ;    cdev - logical dos drive number of the file
                             <1>       ; OUTPUTS ->
                             <1>       ;    u.count - cleared
                             <1>       ;    u.nread - accumulates total bytes passed back
                             <1>       ;
                             <1>       ; ((EAX)) input/output
                             <1>       ; (Retro UNIX Prototype : 14/12/2012 - 01/03/2013, UNIXCOPY.ASM)
                             <1>         ; ((Modified registers: edx, ebx, ecx, esi, edi))
                             <1> 
0000E2AF 31D2                <1>       xor    edx, edx ; 0
0000E2B1 8915[8C030300]      <1>       mov    [u.nread], edx ; 0
0000E2B7 668915[C4030300]    <1>       mov    [u.pcount], dx ; 19/05/2015
0000E2BE 3915[88030300]      <1>       cmp    [u.count], edx ; 0
0000E2C4 7701                <1>       ja     short readi_1
0000E2C6 C3                  <1>       retn
                             <1> readi_1:
                             <1> dskr:
                             <1>       ; 01/05/2016
                             <1>       ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 24/05/2015 - 12/10/2015 (Retro UNIX 386 v1)
                             <1>       ; 26/04/2013 - 03/08/2013 (Retro UNIX 8086 v1)
                             <1> dskr_0:
0000E2C7 8B15[55040300]      <1>         mov  edx, [i.size]
0000E2CD 8B1D[74030300]      <1>       mov    ebx, [u.fofp]
0000E2D3 2B13                <1>       sub    edx, [ebx]
0000E2D5 7647                <1>       jna    short dskr_4
                             <1>       ;
0000E2D7 50                  <1>       push   eax ; 01/05/2016
0000E2D8 3B15[88030300]      <1>       cmp     edx, [u.count]
0000E2DE 7306                <1>       jnb    short dskr_1
0000E2E0 8915[88030300]      <1>       mov    [u.count], edx
                             <1> dskr_1:
                             <1>       ; EAX = First Cluster
                             <1>       ; [Current_Drv] = Physical drive number
0000E2E6 E83B000000          <1>       call   mget_r
                             <1>       ; NOTE: in 'mget_r', relevant sector will be read in buffer
                             <1>       ; if it is not already in buffer !
0000E2EB BB[8C050300]        <1>       mov    ebx, readi_buffer
0000E2F0 803D[C6030300]00    <1>       cmp    byte [u.kcall], 0 ; the caller is 'namei' sign (=1)
0000E2F7 770F                <1>       ja     short dskr_3   ; zf=0 -> the caller is 'namei'
0000E2F9 66833D[C4030300]00  <1>       cmp    word [u.pcount], 0
0000E301 7705                <1>       ja     short dskr_3
                             <1> dskr_2:
                             <1>       ; [u.base] = virtual address to transfer (as destination address)
0000E303 E894010000          <1>       call   trans_addr_w ; translate virtual address to physical (w)
                             <1> dskr_3:
                             <1>       ; EBX (r5) = system (I/O) buffer address -physical-
0000E308 E8F7010000          <1>       call   sioreg
0000E30D 87F7                <1>       xchg   esi, edi
                             <1>       ; EDI = file (user data) offset
                             <1>       ; ESI = sector (I/O) buffer offset
                             <1>       ; ECX = byte count
0000E30F F3A4                <1>       rep    movsb
                             <1>       ; eax = remain bytes in buffer
                             <1>         ;       (check if remain bytes in the buffer > [u.pcount])
0000E311 09C0                <1>       or     eax, eax
0000E313 75EE                <1>       jnz    short dskr_2 ; (page end before system buffer end!)
0000E315 58                  <1>       pop    eax  ; (first cluster number)
0000E316 390D[88030300]      <1>       cmp    [u.count], ecx ; 0
0000E31C 77A9                <1>       ja     short dskr_0
                             <1> dskr_4:
0000E31E C605[C6030300]00    <1>       mov    byte [u.kcall], 0
0000E325 C3                  <1>       retn
                             <1> 
                             <1> mget_r:
                             <1>       ; 24/10/2016
                             <1>       ; 22/10/2016
                             <1>       ; 12/10/2016
                             <1>       ; 29/04/2016
                             <1>       ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 03/06/2015 (Retro UNIX 386 v1, 'mget', u.5s)
                             <1>       ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
```

```
                               <1>        ;
                               <1>        ; Get existing or (allocate) a new disk block for file
                               <1>        ;
                               <1>        ; INPUTS ->
                               <1>        ;    [u.fofp] = file offset pointer
                               <1>        ;    EAX = First Cluster
                               <1>        ;    [cdev] = Logical dos drive number
                               <1>        ;    ([u.off] = file offset)
                               <1>        ; OUTPUTS ->
                               <1>        ;    EAX = logical sector number
                               <1>        ;    ESI = Logical Dos Drive Description Table address
                               <1>        ;
                               <1>        ; Modified registers: EDX, EBX, ECX, ESI, EDI
                               <1>
0000E326 8B35[74030300]        <1>        mov     esi, [u.fofp]
0000E32C 8B1E                  <1>        mov    ebx, [esi] ; (u.off)
                               <1>
0000E32E 29C9                  <1>        sub    ecx, ecx
0000E330 8A2D[46030300]        <1>        mov    ch, [cdev]
                               <1>
0000E336 BE00010900            <1>        mov    esi, Logical_DOSDisks
0000E33B 01CE                  <1>        add    esi, ecx
                               <1>
0000E33D 380D[70650100]        <1>        cmp    [readi.valid], cl ; 0
0000E343 7649                  <1>        jna    short mget_r_0
                               <1>
0000E345 3A2D[71650100]        <1>        cmp    ch, [readi.drv]
0000E34B 7541                  <1>        jne    short mget_r_0
                               <1>
0000E34D 3B05[84650100]        <1>        cmp    eax, [readi.fclust]
0000E353 7565                  <1>        jne    short mget_r_3
                               <1>
0000E355 89D8                  <1>        mov    eax, ebx ; file offset
0000E357 668B0D[78650100]      <1>        mov    cx, [readi.bpc]
0000E35E 41                    <1>        inc    ecx ; <= 65536
0000E35F 29D2                  <1>        sub    edx, edx
0000E361 F7F1                  <1>        div    ecx
                               <1>
0000E363 8B3D[80650100]        <1>        mov    edi, [readi.c_index] ; cluster index
                               <1>
0000E369 39F8                  <1>        cmp    eax, edi
0000E36B 757A                  <1>          jne     short mget_r_4  ; (*)
                               <1>
                               <1>        ; edx = byte offset in cluster (<= 65535)
0000E36D 668915[7A650100]      <1>        mov    [readi.offset], dx
0000E374 66C1EA09              <1>        shr    dx, 9 ; / 512
0000E378 8815[73650100]        <1>        mov    [readi.s_index], dl ; sector index in cluster (0 to spc -1)
                               <1>
0000E37E A1[7C650100]          <1>        mov    eax, [readi.cluster]  ; > 0 if [readi.valid] = 1
0000E383 8B15[88650100]        <1>        mov    edx, [readi.fs_index]
0000E389 E99A000000            <1>          jmp     mget_r_7
                               <1>
                               <1> mget_r_0:
0000E38E 882D[71650100]        <1>        mov    [readi.drv], ch ; physical drive number
0000E394 807E0300              <1>        cmp    byte [esi+LD_FATType], 0
0000E398 7707                  <1>        ja     short mget_r_1
0000E39A 8A4E12                <1>        mov    cl, [esi+LD_FS_BytesPerSec+1]
0000E39D D0E9                  <1>        shr    cl, 1 ;  ; 1 for 512 bytes, 4 for 2048 bytes
0000E39F EB03                  <1>        jmp    short mget_r_2
                               <1> mget_r_1:
0000E3A1 8A4E13                <1>        mov    cl, [esi+LD_BPB+BPB_SecPerClust]
                               <1> mget_r_2:
0000E3A4 880D[72650100]        <1>        mov    [readi.spc], cl  ; sectors per cluster
                               <1>        ; NOTE: readi bytes per sector value is always 512 !
0000E3AA 66C1E109              <1>        shl    cx, 9 ; * 512
0000E3AE 6649                  <1>        dec    cx ; bytes per cluster - 1
0000E3B0 66890D[78650100]      <1>        mov    [readi.bpc], cx
0000E3B7 6629C9                <1>        sub    cx, cx
                               <1> mget_r_3:
0000E3BA A3[84650100]          <1>        mov    [readi.fclust], eax ; first cluster (or FDT address)
0000E3BF 880D[70650100]        <1>        mov    [readi.valid], cl ; 0
                               <1>        ;mov   [readi.s_index], cl ; 0
                               <1>        ;mov   [readi.offset], cx ; 0
0000E3C5 890D[80650100]        <1>        mov    [readi.c_index], ecx ; 0
0000E3CB 890D[7C650100]        <1>        mov    [readi.cluster], ecx ; 0
0000E3D1 890D[74650100]        <1>        mov    [readi.sector], ecx ; 0
                               <1>
0000E3D7 89D8                  <1>        mov    eax, ebx ; file offset
0000E3D9 668B0D[78650100]      <1>        mov    cx, [readi.bpc]
0000E3E0 41                    <1>        inc    ecx ; <= 65536
0000E3E1 29D2                  <1>        sub    edx, edx
0000E3E3 F7F1                  <1>        div    ecx
                               <1>        ;mov   edi, [readi.c_index] ; previous cluster index
0000E3E5 29FF                  <1>        sub    edi, edi
                               <1> mget_r_4:
0000E3E7 A3[80650100]          <1>        mov    [readi.c_index], eax ; cluster index
                               <1>        ; edx = byte offset in cluster (<= 65535)
0000E3EC 668915[7A650100]      <1>        mov    [readi.offset], dx
0000E3F3 66C1EA09              <1>        shr    dx, 9 ; / 512
0000E3F7 8815[73650100]        <1>        mov    [readi.s_index], dl ; sector index in cluster (0 to spc -1)
                               <1>
0000E3FD 89C1                  <1>        mov    ecx, eax ; current cluster index
0000E3FF A1[84650100]          <1>        mov    eax, [readi.fclust]
0000E404 09C9                  <1>        or     ecx, ecx ; cluster index
0000E406 741B                  <1>        jz     short mget_r_6
                               <1>
0000E408 39CF                  <1>        cmp    edi, ecx
0000E40A 7710                  <1>        ja     short mget_r_5 ; old cluster index is higher
0000E40C 8B15[7C650100]        <1>        mov    edx, [readi.cluster]
0000E412 21D2                  <1>        and    edx, edx
0000E414 7406                  <1>        jz     short mget_r_5
                               <1>        ; valid 'readi' parameters (*)
0000E416 89D0                  <1>        mov    eax, edx
```

```
0000E418 29F9                <1>       sub   ecx, edi
0000E41A 740C                <1>       jz    short mget_r_7
                             <1> mget_r_5:
                             <1>       ; EAX = Beginning cluster
                             <1>       ; EDX = Sector index in disk/file section
                             <1>       ;      (Only for SINGLIX file system!)
                             <1>       ; ECX = Cluster sequence number after the beginning cluster
                             <1>       ; ESI = Logical DOS Drive Description Table address
0000E41C E836E1FFFF          <1>       call  get_cluster_by_index
0000E421 724E                <1>       jc    short mget_r_err
                             <1>       ; EAX = Cluster number
                             <1> mget_r_6:
0000E423 A3[7C650100]        <1>       mov   [readi.cluster], eax ; FDT number for Singlix File System
                             <1> mget_r_7:
0000E428 807E0300            <1>       cmp   byte [esi+LD_FATType], 0
0000E42C 765F                <1>       jna   short mget_r_12
                             <1>
0000E42E 83E802              <1>       sub   eax, 2
0000E431 0FB615[72650100]    <1>       movzx edx, byte [readi.spc]
0000E438 F7E2                <1>       mul   edx
                             <1>
0000E43A 034668              <1>       add   eax, [esi+LD_DATABegin]
0000E43D 8A15[73650100]      <1>       mov   dl, [readi.s_index]
0000E443 01D0                <1>       add   eax, edx
                             <1> mget_r_8:
                             <1>       ; eax = logical sector number
0000E445 803D[70650100]00    <1>       cmp   byte [readi.valid], 0
0000E44C 7608                <1>       jna   short mget_r_9
0000E44E 3B05[74650100]      <1>       cmp   eax, [readi.sector]
0000E454 7436                <1>       je    short mget_r_11 ; sector is already in 'readi' buffer
                             <1> mget_r_9:
0000E456 A3[74650100]        <1>       mov   [readi.sector], eax
0000E45B BB[8C050300]        <1>       mov   ebx, readi_buffer ; buffer address
0000E460 B901000000          <1>       mov   ecx, 1
                             <1>       ; 29/04/2016
                             <1>       ;xor  dl, dl
                             <1>
                             <1>       ; EAX = Logical sector number
                             <1>       ; ECX = Sector count
                             <1>       ; EBX = Buffer address
                             <1>       ; (EDX = 0)
                             <1>       ; ESI = Logical DOS drive description table address
                             <1>
0000E465 E86E130000          <1>       call  disk_read
0000E46A 7314                <1>       jnc   short mget_r_10
                             <1>
                             <1>       ; 22/10/2016 (15h -> 17)
0000E46C B811000000          <1>       mov   eax, 17 ; Drive not ready or read error !
                             <1> mget_r_err:
0000E471 A3[C8030300]        <1>       mov   [u.error], eax
                             <1>       ; 12/10/2016
0000E476 A3[64030300]        <1>       mov   [u.r0], eax
0000E47B E93EE2FFFF          <1>       jmp   error
                             <1> mget_r_10:
0000E480 C605[70650100]01    <1>       mov   byte [readi.valid], 1 ; 24/10/2016
0000E487 A1[74650100]        <1>       mov   eax, [readi.sector]
                             <1> mget_r_11:
0000E48C C3                  <1>       retn
                             <1> mget_r_12:
                             <1>       ; EAX = FDT number
                             <1>       ; EDX = Sector index from FDT sector (0,1,2,3,4...)
0000E48D 40                  <1>       inc   eax ; the first data sector in FS disk section
0000E48E 8915[88650100]      <1>       mov   [readi.fs_index], edx
0000E494 01D0                <1>       add   eax, edx
0000E496 EBAD                <1>       jmp   short mget_r_8
                             <1>
                             <1> trans_addr_r:
                             <1>       ; 12/10/2016
                             <1>       ; 02/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; Translate virtual address to physical address
                             <1>       ; for reading from user's memory space
                             <1>       ; 04/06/2015 - 18/10/2015 (Retro UNIX 386 v1)
                             <1>
0000E498 31D2                <1>       xor   edx, edx ; 0 (read access sign)
0000E49A EB04                <1>       jmp   short trans_addr_rw
                             <1>
                             <1> trans_addr_w:
                             <1>       ; 12/10/2016
                             <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; Translate virtual address to physical address
                             <1>       ; for writing to user's memory space
                             <1>       ; 04/06/2015 - 18/10/2015 (Retro UNIX 386 v1)
                             <1>
0000E49C 29D2                <1>       sub   edx, edx
0000E49E FEC2                <1>       inc   dl ; 1 (write access sign)
                             <1> trans_addr_rw:
0000E4A0 50                  <1>       push  eax
0000E4A1 53                  <1>       push  ebx
0000E4A2 52                  <1>       push  edx ; r/w sign (in DL)
                             <1>       ;
0000E4A3 8B1D[84030300]      <1>       mov   ebx, [u.base]
0000E4A9 E8E16DFFFF          <1>       call  get_physical_addr ; get physical address
0000E4AE 730F                <1>       jnc   short passc_0
0000E4B0 A3[C8030300]        <1>       mov   [u.error], eax
0000E4B5 A3[64030300]        <1>       mov   [u.r0], eax ; 12/10/2016
                             <1>       ;pop  edx
                             <1>       ;pop  ebx
                             <1>       ;pop  eax
0000E4BA E9FFE1FFFF          <1>       jmp   error
                             <1> passc_0:
0000E4BF F6C202              <1>       test  dl, PTE_A_WRITE ; writable page
0000E4C2 5A                  <1>       pop   edx
0000E4C3 751C                <1>       jnz   short passc_1
```

```
                             <1>
0000E4C5 20D2                <1>       and    dl, dl
0000E4C7 7418                <1>       jz     short passc_1
                             <1>       ; read only (duplicated) page -must be copied to a new page-
                             <1>       ; EBX = linear address
0000E4C9 51                  <1>       push   ecx
0000E4CA E8596AFFFF          <1>       call   copy_page
0000E4CF 59                  <1>       pop    ecx
0000E4D0 721E                <1>       jc     short passc_2
0000E4D2 50                  <1>       push   eax ; physical address of the new/allocated page
0000E4D3 E8E16CFFFF          <1>       call   add_to_swap_queue
0000E4D8 58                  <1>       pop    eax
0000E4D9 81E3FF0F0000        <1>       and    ebx, PAGE_OFF ; 0FFFh
                             <1>       ;mov   ecx, PAGE_SIZE
                             <1>       ;sub   ecx, ebx
0000E4DF 01D8                <1>       add    eax, ebx
                             <1> passc_1:
0000E4E1 A3[C0030300]        <1>       mov    [u.pbase], eax ; physical address
0000E4E6 66890D[C4030300]    <1>       mov    [u.pcount], cx ; remain byte count in page (1-4096)
0000E4ED 5B                  <1>       pop    ebx
0000E4EE 58                  <1>       pop    eax
0000E4EF C3                  <1>       retn
                             <1> passc_2:
0000E4F0 B804000000          <1>       mov    eax, ERR_MINOR_IM ; "Insufficient memory !" error
0000E4F5 A3[64030300]        <1>       mov    [u.r0], eax ; 12/10/2016
0000E4FA A3[C8030300]        <1>       mov    dword [u.error], eax
                             <1>       ;pop   ebx
                             <1>       ;pop   eax
0000E4FF E9BAE1FFFF          <1>       jmp    error
                             <1>
                             <1> sioreg:
                             <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 19/05/2015 - 25/07/2015 (Retro UNIX 386 v1)
                             <1>       ; 12/03/2013 - 22/07/2013 (Retro UNIX 8086 v1)
                             <1>       ; INPUTS ->
                             <1>       ;     EBX = system buffer (data) address (r5)
                             <1>       ;     [u.fofp] = pointer to file offset pointer
                             <1>       ;     [u.base] = virtual address of the user buffer
                             <1>       ;     [u.pbase] = physical address of the user buffer
                             <1>       ;     [u.count] = byte count
                             <1>       ;     [u.pcount] = byte count within page frame
                             <1>       ; OUTPUTS ->
                             <1>       ;     ESI = user data offset (r1)
                             <1>       ;     EDI = system (I/O) buffer offset (r2)
                             <1>       ;     ECX = byte count (r3)
                             <1>       ;     EAX = remain bytes after byte count within page frame
                             <1>       ;      (If EAX > 0, transfer will continue from the next page)
                             <1>         ;
                             <1>       ; ((Modified registers:  EDX))
                             <1>
0000E504 8B35[74030300]      <1>         mov     esi, [u.fofp]
0000E50A 8B3E                <1>         mov     edi, [esi]
0000E50C 89F9                <1>       mov    ecx, edi
0000E50E 81C900FEFFFF        <1>       or     ecx, 0FFFFFE00h
0000E514 81E7FF010000        <1>       and    edi, 1FFh
0000E51A 01DF                <1>       add    edi, ebx ; EBX = system buffer (data) address
0000E51C F7D9                <1>       neg    ecx
0000E51E 3B0D[88030300]      <1>       cmp    ecx, [u.count]
0000E524 7606                <1>       jna    short sioreg_0
0000E526 8B0D[88030300]      <1>       mov    ecx, [u.count]
                             <1> sioreg_0:
0000E52C 803D[C6030300]00    <1>       cmp    byte [u.kcall], 0
0000E533 7613                <1>       jna    short sioreg_1
                             <1>        ; the caller is 'mkdir' or 'namei'
0000E535 A1[84030300]        <1>       mov    eax, [u.base]
0000E53A A3[C0030300]        <1>       mov    [u.pbase], eax ; physical address = virtual address
0000E53F 66890D[C4030300]    <1>       mov    word [u.pcount], cx ; remain bytes in buffer (1 sector)
0000E546 EB0B                <1>       jmp    short sioreg_2
                             <1> sioreg_1:
0000E548 0FB715[C4030300]    <1>       movzx  edx, word [u.pcount]
0000E54F 39D1                <1>       cmp    ecx, edx
0000E551 772A                <1>       ja     short sioreg_4 ; transfer count > [u.pcount]
                             <1> sioreg_2: ; 2:
0000E553 31C0                <1>       xor    eax, eax
                             <1> sioreg_3:
0000E555 010D[8C030300]      <1>       add    [u.nread], ecx
0000E55B 290D[88030300]      <1>       sub    [u.count], ecx
0000E561 010D[84030300]      <1>       add    [u.base], ecx
0000E567 010E                <1>         add  [esi], ecx
0000E569 8B35[C0030300]      <1>       mov    esi, [u.pbase]
0000E56F 66290D[C4030300]    <1>       sub    [u.pcount], cx
0000E576 010D[C0030300]      <1>       add    [u.pbase], ecx
0000E57C C3                  <1>         retn
                             <1> sioreg_4:
                             <1>       ; transfer count > [u.pcount]
                             <1>       ; (ecx > edx)
0000E57D 89C8                <1>       mov    eax, ecx
0000E57F 29D0                <1>       sub    eax, edx ; remain bytes for 1 sector (block) transfer
0000E581 89D1                <1>       mov    ecx, edx ; current transfer count = [u.pcount]
0000E583 EBD0                <1>       jmp    short sioreg_3
                             <1>
                             <1> tswitch: ; Retro UNIX 386 v1
                             <1> tswap:
                             <1>       ; 16/01/2017
                             <1>       ; 21/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 10/05/2015 - 01/09/2015 (Retro UNIX 386 v1)
                             <1>       ; 14/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
                             <1>       ; time out swap, called when a user times out.
                             <1>       ; the user is put on the low priority queue.
                             <1>       ; This is done by making a link from the last user
                             <1>       ; on the low priority queue to him via a call to 'putlu'.
                             <1>       ; then he is swapped out.
                             <1>
                             <1>       ; TRDOS 386 (TRDOS v2.0) modification ->  ** 21/05/2016 **
                             <1>       ;     * when a high priority (event) process will be stopped
                             <1>       ;      (swapped out, swithched out/off), 'tswap/tswitch' will
                             <1>       ;      not add it to a run queue.
                             <1>       ;      /// What for: Process may be already in a run queue,
                             <1>       ;      it is unspeficied state because process might be started
                             <1>       ;      by a timer event which does not regard previous priority
                             <1>       ;      level and run queue of the process (for fast executing!).
                             <1>       ;      After the 'run for event', process will be sequenced
                             <1>       ;      to run by it's actual run queue. ///
                             <1>       ;
                             <1>       ; Retro UNIX 386 v1 modification ->
                             <1>       ;       swap (software task switch) is performed by changing
                             <1>       ;      user's page directory (u.pgdir) instead of segment change
                             <1>       ;      as in Retro UNIX 8086 v1.
                             <1>       ;
                             <1>       ; RETRO UNIX 8086 v1 modification ->
                             <1>       ;       'swap to disk' is replaced with 'change running segment'
                             <1>       ;      according to 8086 cpu (x86 real mode) architecture.
                             <1>       ;      pdp-11 was using 64KB uniform memory while IBM PC
                             <1>       ;      compatibles was using 1MB segmented memory
                             <1>       ;      in 8086/8088 times.
                             <1>       ;
                             <1>       ; INPUTS ->
                             <1>       ;    u.uno - users process number
                             <1>       ;    runq+4 - lowest priority queue
                             <1>       ; OUTPUTS ->
                             <1>       ;    r0 - users process number
                             <1>       ;    r2 - lowest priority queue address
                             <1>       ;
                             <1>       ; ((AX = R0, BX = R2)) output
                             <1>       ; ((Modified registers: EDX, EBX, ECX, ESI, EDI))
                             <1>       ;
                             <1>
                             <1>       NOTE:
                             <1>       ;* [u.pri] priority level is specified by run queue which is process
                             <1>       ;  comes to run from.
                             <1>       ;* Initial [u.pri] is 1 ('normal/regular') for programs
                             <1>       ;  (which are launched by MainProg or 'sysexec'), it is changed
                             <1>       ;  to 2 ('high') by timer event, if program uses 'systimer' system call.
                             <1>       ;* Program (Process) also can change it's running priority
                             <1>       ;  from 1 to 0 or up to 2 by using 'syspri' system call; but,
                             <1>       ;  if program selects priority level 2 (high) for running, next time
                             <1>       ;  it is reduced to 1 (normal/regular) because 'syspri' adds this
                             <1>       ;  program to 'run for normal' queue while running duration is a bit
                             <1>       ;  protected from swap/switch out immediate, behalf of other high
                             <1>       ;  priority process in sequence. Program (with high priority) will not
                             <1>       ;  be swapped/switched out (by timer event) before it's time quantum
                             <1>       ;  will be elapsed, but, this will be temporary if program is not using
                             <1>       ;  timer event function.
                             <1>
                             <1>       ;For example:
                             <1>       ;If a process frequently gets a timer event, it runs at high priority
                             <1>       ;level but when it returns from running it returns to actual run queue,
                             <1>       ;not to 'run for event' queue again.
                             <1>       ;'tswap' will not change the sequence at return/stop(swap out) stage.
                             <1>       ;But if priority level not high (=2, 'run for event'), 'tswap/tswitch'
                             <1>       ;will add the stopping process to relevant run queue according to
                             <1>       ;[u.pri] priority level.
                             <1>
                             <1>       ; 16/01/2017
0000E585 BB[54030300]        <1>       mov    ebx, runq+2  ; 'runq_normal' ; normal/regular priority
                             <1>       ; 21/05/2016
                             <1>       ;cmp   byte [u.pri], 2     ; high priority (run for event) ?
                             <1>       ;jnb   short swap
                             <1>       ; 16/01/2017
                             <1>       ; (Normal and also high/event priority processes will be added to
                             <1>       ; normal priority run queue for ensuring circular running sequence!)
                             <1>       ; (Timer interrupt or 'syspri' system call may change priority and run
                             <1>       ; queue to high/event level.)
0000E58A 803D[A9030300]00    <1>       cmp    byte [u.pri], 0
0000E591 7702                <1>       ja     short tswap_1; normal priority run queue
                             <1>       ;
0000E593 43                  <1>       inc    ebx
0000E594 43                  <1>       inc    ebx          ; runq+4, 'runq_background', low priority
                             <1> tswap_1:
0000E595 A0[B3030300]        <1>       mov    al, [u.uno]
                             <1>                    ; movb u.uno,r1 / move users process number to r1
                             <1>              ; mov  $runq+4,r2
                             <1>                    ; / move lowest priority queue address to r2
                             <1>              ; ebx = run queue
0000E59A E8FE000000          <1>       call   putlu
                             <1>              ; jsr r0,putlu / create link from last user on Q to
                             <1>                          ; / u.uno's user
                             <1>
                             <1> switch: ; Retro UNIX 386 v1
                             <1> swap:
                             <1>       ; 02/01/2017
                             <1>       ; 21/05/2016
                             <1>       ; 20/05/2016
                             <1>       ; 02/05/2016
                             <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 10/05/2015 - 02/09/2015 (Retro UNIX 386 v1)
                             <1>       ; 14/04/2013 - 08/03/2014 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 'swap' is routine that controls the swapping of processes
                             <1>       ; in and out of core.
                             <1>       ;
                             <1>       ; TRDOS 386 (TRDOS v2.0) modification ->  ** 20/05/2016 **
                             <1>       ;     * 3 different priority level is applied
                             <1>       ;      (just as original unix v1)
                             <1>       ;      1) high priority (event) run queue, 'runq_event'
                             <1>       ;      2) normal priority (regular) run queue, 'runq_normal'
                             <1>       ;      3) low priority (background) run queue, 'runq_backgroud'
                             <1>       ;      'swap' code will run a process which has max. priority
                             <1>         ;       (for earliest event at first)
                             <1>       ;
                             <1>       ; Retro UNIX 386 v1 modification ->
                             <1>       ;       swap (software task switch) is performed by changing
                             <1>       ;      user's page directory (u.pgdir) instead of segment change
                             <1>       ;      as in Retro UNIX 8086 v1.
                             <1>       ;
                             <1>       ; RETRO UNIX 8086 v1 modification ->
                             <1>       ;       'swap to disk' is replaced with 'change running segment'
                             <1>       ;      according to 8086 cpu (x86 real mode) architecture.
                             <1>       ;      pdp-11 was using 64KB uniform memory while IBM PC
                             <1>       ;      compatibles was using 1MB segmented memory
                             <1>       ;      in 8086/8088 times.
                             <1>       ;
                             <1>       ; INPUTS ->
                             <1>       ;    runq table - contains processes to run.
                             <1>       ;    p.link - contains next process in line to be run.
                             <1>       ;    u.uno - process number of process in core
                             <1>       ;    s.stack - swap stack used as an internal stack for swapping.
                             <1>       ; OUTPUTS ->
                             <1>       ;    (original unix v1 -> present process to its disk block)
                             <1>       ;    (original unix v1 -> new process into core ->
                             <1>       ;        Retro Unix 8086 v1 -> segment registers changed
                             <1>       ;        for new process)
                             <1>       ;    u.quant = 3 (Time quantum for a process)
                             <1>       ;      ((INT 1Ch count down speed -> 18.2 times per second)
                             <1>       ;    RETRO UNIX 8086 v1 will use INT 1Ch (18.2 times per second)
                             <1>       ;       for now, it will swap the process if there is not
                             <1>       ;       a keyboard event (keystroke) (Int 15h, function 4Fh)
                             <1>       ;       or will count down from 3 to 0 even if there is a
                             <1>       ;         keyboard event locking due to repetitive key strokes.
                             <1>       ;       u.quant will be reset to 3 for RETRO UNIX 8086 v1.
                             <1>       ;
                             <1>       ; ((Modified registers: EAX, EDX, EBX, ECX, ESI, EDI))
                             <1>
                             <1>       ;NOTE:
                             <1>       ;High priority queue is the first for selecting a process to run.
                             <1>       ;If there is not a process in high priority level run queue,
                             <1>       ;a process in normal priority run queue will be selected
                             <1>       ;or a proces in low priority run queue will be selected if normal
                             <1>       ;priority level run queue is empty.
                             <1>
                             <1>       ; 21/05/2016 -(3 priority levels, 3 run queues)
0000E59F BE[52030300]        <1>       mov    esi, runq ; 'runq_event' ; high priority, 'run for event'
0000E5A4 C605[CC650100]03    <1>       mov    byte [priority], 3 ; high priority + 1
0000E5AB 31DB                <1>       xor    ebx, ebx ; 02/01/2017
                             <1> swap_0: ; 1: / search runq table for highest priority process
0000E5AD 66AD                <1>       lodsw  ; mov ax, [esi], add esi+2
                             <1>       ;xor   ebx, ebx ; 02/05/2016
0000E5AF 6621C0              <1>       and    ax, ax ; are there any processes to run in this Q entry
0000E5B2 750E                <1>       jnz    short swap_2
                             <1>       ; 21/05/2026
                             <1>       ; runq_normal = runq+2, runq_background = runq+4
0000E5B4 FE0D[CC650100]      <1>       dec    byte [priority] ; 3 -> 3, 2 -> 1, 1-> 0
0000E5BA 75F1                <1>       jnz    short swap_0
                             <1>       ;cmp   esi, runq+6  ; if zero compare address to end of table
                             <1>       ;jb    short swap_0 ; if not at end, go back
                             <1> swap_1:
                             <1>       ; 02/05/2016
                             <1>       ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)
                             <1>       ; No user process to run...
                             <1>       ; Run the kernel process... MainProg: Internal Command Interpreter
0000E5BC FEC0                <1>       inc    al ; mov al, 1  ; process number of MainProg
0000E5BE FEC3                <1>       inc    bl ; mov bl, al ; 1
0000E5C0 EB1E                <1>       jmp    short swap_4
                             <1> swap_2:
                             <1>       ; 21/05/2016
0000E5C2 FE0D[CC650100]      <1>       dec    byte [priority] ; priority level of present user/process
                             <1>                                  ; 0, 1, 2
0000E5C8 4E                  <1>       dec    esi
0000E5C9 4E                  <1>         dec     esi
                             <1>       ;
0000E5CA 88C3                <1>       mov    bl, al
0000E5CC 38E0                <1>       cmp    al, ah ; is there only 1 process in the queue to be run
0000E5CE 740A                <1>       je     short swap_3 ; yes
0000E5D0 8AA3[9F000300]      <1>       mov    ah, [ebx+p.link-1]
0000E5D6 8826                <1>              mov    [esi], ah ; move next process in line into run queue
0000E5D8 EB06                <1>       jmp    short swap_4
                             <1> swap_3:
0000E5DA 6631D2              <1>       xor    dx, dx
0000E5DD 668916              <1>       mov    [esi], dx ; zero the entry; no processes on the Q
                             <1> swap_4:
0000E5E0 8A25[B3030300]      <1>       mov    ah, [u.uno]
0000E5E6 38C4                <1>       cmp    ah, al ; is this process the same as the process in core?
0000E5E8 743B                <1>              je     short swap_8 ; yes, don't have to swap
0000E5EA 08E4                <1>       or     ah, ah  ; is the process # = 0
0000E5EC 740D                <1>              jz     short swap_6 ; 'sysexit'
                             <1>       ;cmp   ah, al ;is this process the same as the process in core?
                             <1>              ;je    short swap_8 ; yes, don't have to swap
0000E5EE 8925[60030300]      <1>       mov    [u.usp], esp ; return  address for 'syswait' & 'sleep'
0000E5F4 E834000000          <1>       call   wswap   ; write out core to disk
0000E5F9 EB1C                <1>       jmp    short swap_7
                             <1> swap_6:
                             <1>       ; Deallocate memory pages belong to the process
                             <1>       ; which is being terminated.
                             <1>       ; (Retro UNIX 386 v1 modification !)
                             <1>       ;
0000E5FB 53                  <1>       push   ebx
0000E5FC A1[B8030300]        <1>       mov    eax, [u.pgdir]  ; page directory of the process
0000E601 8B1D[BC030300]      <1>       mov    ebx, [u.ppgdir] ; page directory of the parent process
0000E607 E8A766FFFF          <1>       call   deallocate_page_dir
```

```
0000E60C A1[B4030300]        <1>       mov    eax, [u.upage] ; 'user' structure page of the process
0000E611 E84267FFFF          <1>       call   deallocate_page
0000E616 5B                  <1>       pop    ebx
                             <1> swap_7:
0000E617 C0E302              <1>       shl    bl, 2 ; * 4
0000E61A 8B83[BC000300]      <1>       mov    eax, [ebx+p.upage-4] ; the 'u' page of the new process
0000E620 E840000000          <1>       call   rswap ; read new process into core
                             <1> swap_8:
                             <1>       ; Retro UNIX  8086 v1 modification !
0000E625 C605[A8030300]04    <1>       mov    byte [u.quant], time_count
0000E62C C3                  <1>       retn
                             <1>
                             <1> wswap:  ; < swap out, swap to disk >
                             <1>       ; 28/02/2017 (fnsave)
                             <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 09/05/2015 (Retro UNIX 386 v1)
                             <1>       ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
                             <1>       ; 'wswap' writes out the process that is in core onto its
                             <1>       ; appropriate disk area.
                             <1>       ;
                             <1>       ; Retro UNIX 386 v1 modification ->
                             <1>       ;       User (u) structure content and the user's register content
                             <1>       ;      will be copied to the process's/user's UPAGE (a page for
                             <1>       ;      saving 'u' structure and user registers for task switching).
                             <1>       ;      u.usp - points to kernel stack address which contains
                             <1>       ;             user's registers while entering system call.
                             <1>       ;      u.sp  - points to kernel stack address
                             <1>       ;             to return from system call -for IRET-.
                             <1>       ;      [u.usp]+32+16 = [u.sp]
                             <1>       ;      [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
                             <1>       ;             edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
                             <1>       ;
                             <1>       ; Retro UNIX 8086 v1 modification ->
                             <1>       ;       'swap to disk' is replaced with 'change running segment'
                             <1>       ;      according to 8086 cpu (x86 real mode) architecture.
                             <1>       ;      pdp-11 was using 64KB uniform memory while IBM PC
                             <1>       ;      compatibles was using 1MB segmented memory
                             <1>       ;      in 8086/8088 times.
                             <1>       ;
                             <1>       ; INPUTS ->
                             <1>       ;    u.break - points to end of program
                             <1>       ;    u.usp - stack pointer at the moment of swap
                             <1>       ;    core - beginning of process program
                             <1>       ;    ecore - end of core
                             <1>       ;    user - start of user parameter area
                             <1>       ;    u.uno - user process number
                             <1>       ;    p.dska - holds block number of process
                             <1>       ; OUTPUTS ->
                             <1>       ;    swp I/O queue
                             <1>       ;    p.break - negative word count of process
                             <1>       ;    r1 - process disk address
                             <1>       ;    r2 - negative word count
                             <1>       ;
                             <1>       ; RETRO UNIX 8086 v1 input/output:
                             <1>       ;
                             <1>       ; INPUTS ->
                             <1>       ;    u.uno - process number (to be swapped out)
                             <1>       ; OUTPUTS ->
                             <1>       ;    none
                             <1>       ;
                             <1>       ;   ((Modified registers: ECX, ESI, EDI))
                             <1>       ;
                             <1>
                             <1>       ; 28/02/2017
                             <1>       ;cmp   byte [multi_tasking], 0  ; Musti tasking mode ?
                             <1>       ;jna   short wswp
0000E62D 803D[DA030300]00    <1>       cmp    byte [u.fpsave], 0 ; 28/02/2017
0000E634 7606                <1>       jna    short wswp
0000E636 DD35[DC030300]      <1>       fnsave [u.fpregs] ; save floating point registers (94 bytes)
                             <1> wswp:
0000E63C 8B3D[B4030300]      <1>       mov    edi, [u.upage] ; process's user (u) structure page addr
0000E642 B938000000          <1>       mov    ecx, (U_SIZE + 3) / 4
0000E647 BE[5C030300]        <1>       mov    esi, user ; active user (u) structure
0000E64C F3A5                <1>       rep    movsd
                             <1>       ;
0000E64E 8B35[60030300]      <1>       mov    esi, [u.usp] ; esp (system stack pointer,
                             <1>                          ;      points to user registers)
0000E654 8B0D[5C030300]      <1>       mov    ecx, [u.sp]  ; return address from the system call
                             <1>                          ; (for IRET)
                             <1>                          ; [u.sp] -> EIP (user)
                             <1>                          ; [u.sp+4]-> CS (user)
                             <1>                          ; [u.sp+8] -> EFLAGS (user)
                             <1>                          ; [u.sp+12] -> ESP (user)
                             <1>                          ; [u.sp+16] -> SS (user)
0000E65A 29F1                <1>       sub    ecx, esi     ; required space for user registers
0000E65C 83C114              <1>       add    ecx, 20           ; +5 dwords to return from system call
                             <1>                          ; (for IRET)
0000E65F C1E902              <1>       shr    ecx, 2
0000E662 F3A5                <1>       rep    movsd
0000E664 C3                  <1>       retn
                             <1>
                             <1> rswap:  ; < swap in, swap from disk >
                             <1>       ; 28/02/2017 (frstor)
                             <1>       ; 15/01/2017
                             <1>       ; 14/01/2017
                             <1>       ; 21/05/2016
                             <1>       ; 03/05/2016
                             <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 09/05/2015 - 15/09/2015 (Retro UNIX 386 v1)
                             <1>       ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
                             <1>       ; 'rswap' reads a process whose number is in r1,
                             <1>       ; from disk into core.
                             <1>       ;
```

```
                            <1>       ; Retro UNIX 386 v1 modification ->
                            <1>       ;       User (u) structure content and the user's register content
                            <1>       ;      will be restored from process's/user's UPAGE (a page for
                            <1>       ;      saving 'u' structure and user registers for task switching).
                            <1>       ;      u.usp - points to kernel stack address which contains
                            <1>       ;             user's registers while entering system call.
                            <1>       ;      u.sp  - points to kernel stack address
                            <1>       ;             to return from system call -for IRET-.
                            <1>       ;      [u.usp]+32+16 = [u.sp]
                            <1>       ;      [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
                            <1>       ;             edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
                            <1>       ;
                            <1>       ; RETRO UNIX 8086 v1 modification ->
                            <1>       ;       'swap to disk' is replaced with 'change running segment'
                            <1>       ;      according to 8086 cpu (x86 real mode) architecture.
                            <1>       ;      pdp-11 was using 64KB uniform memory while IBM PC
                            <1>       ;      compatibles was using 1MB segmented memory
                            <1>       ;      in 8086/8088 times.
                            <1>       ;
                            <1>       ; INPUTS ->
                            <1>       ;    r1 - process number of process to be read in
                            <1>       ;    p.break - negative of word count of process
                            <1>       ;    p.dska - disk address of the process
                            <1>       ;    u.emt - determines handling of emt's
                            <1>       ;    u.ilgins - determines handling of illegal instructions
                            <1>       ; OUTPUTS ->
                            <1>       ;    8 = (u.ilgins)
                            <1>       ;    24 = (u.emt)
                            <1>       ;    swp - bit 10 is set to indicate read
                            <1>       ;            (bit 15=0 when reading is done)
                            <1>       ;    swp+2 - disk block address
                            <1>       ;    swp+4 - negative word count
                            <1>       ;      ((swp+6 - address of user structure))
                            <1>       ;
                            <1>       ; RETRO UNIX 8086 v1 input/output:
                            <1>       ;
                            <1>       ; INPUTS ->
                            <1>       ;    AL       - new process number (to be swapped in)
                            <1>       ; OUTPUTS ->
                            <1>       ;    none
                            <1>       ;
                            <1>       ;   ((Modified registers: EAX, ECX, ESI, EDI, ESP))
                            <1>       ;
                            <1>       ; Retro UNIX 386 v1 - modification ! 14/05/2015
0000E665 89C6               <1>       mov    esi, eax  ; process's user (u) structure page addr
0000E667 B938000000         <1>       mov    ecx, (U_SIZE + 3) / 4
0000E66C BF[5C030300]       <1>       mov    edi, user ; active user (u) structure
0000E671 F3A5               <1>       rep    movsd
0000E673 58                 <1>       pop    eax   ; 'rswap' return address
                            <1>       ;
                            <1>       ;cli
0000E674 8B3D[60030300]     <1>       mov    edi, [u.usp] ; esp (system stack pointer,
                            <1>                         ;     points to user registers)
0000E67A 89FC               <1>       mov    esp, edi    ; 14/01/2017
0000E67C 8B0D[5C030300]     <1>       mov    ecx, [u.sp]  ; return address from the system call
                            <1>                         ; (for IRET)
                            <1>                         ; [u.sp] -> EIP (user)
                            <1>                         ; [u.sp+4]-> CS (user)
                            <1>                         ; [u.sp+8] -> EFLAGS (user)
                            <1>                         ; [u.sp+12] -> ESP (user)
                            <1>                         ; [u.sp+16] -> SS (user)
0000E682 29F9               <1>       sub    ecx, edi    ; required space for user registers
0000E684 83C114             <1>       add    ecx, 20          ; +5 dwords to return from system call
                            <1>                         ; (for IRET)
0000E687 C1E902             <1>       shr    ecx, 2
0000E68A F3A5               <1>       rep    movsd
                            <1>       ;mov   esp, [u.usp] ; 15/09/2015
                            <1>       ;sti
                            <1>       ; 28/02/2017
                            <1>       ;cmp   byte [multi_tasking], 0  ; Musti tasking mode ?
                            <1>       ;jna   short rswp_retn
0000E68C 803D[DA030300]00   <1>       cmp    byte [u.fpsave], 0
0000E693 7606               <1>       jna    short rswp_retn
0000E695 DD25[DC030300]     <1>       frstor [u.fpregs] ; restore floating point regs (94 bytes)
                            <1> rswp_retn:
0000E69B 50                 <1>       push   eax   ; 'rswap' return address
0000E69C C3                 <1>       retn
                            <1>
                            <1> putlu:
                            <1>       ; 20/05/2016
                            <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                            <1>       ; 10/05/2015 - 12/09/2015 (Retro UNIX 386 v1)
                            <1>       ; 15/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
                            <1>       ; 'putlu' is called with a process number in r1 and a pointer
                            <1>       ; to lowest priority Q (runq+4) in r2. A link is created from
                            <1>       ; the last process on the queue to process in r1 by putting
                            <1>       ; the process number in r1 into the last process's link.
                            <1>       ;
                            <1>       ; INPUTS ->
                            <1>       ;    r1 - user process number
                            <1>       ;    r2 - points to lowest priority queue
                            <1>       ;    p.dska - disk address of the process
                            <1>       ;    u.emt - determines handling of emt's
                            <1>       ;    u.ilgins - determines handling of illegal instructions
                            <1>       ; OUTPUTS ->
                            <1>       ;    r3 - process number of last process on the queue upon
                            <1>       ;       entering putlu
                            <1>       ;    p.link-1 + r3 - process number in r1
                            <1>       ;    r2 - points to lowest priority queue
                            <1>       ;
                            <1>       ; ((Modified registers: EDX, EBX))
                            <1>       ;
                            <1>       ; / r1 = user process no.; r2 points to lowest priority queue
```

```
                             <1>
                             <1>       ; EBX = r2
                             <1>       ; EAX = r1 (AL=r1b)
                             <1>
                             <1>       ; 20/05/2016
                             <1>       ; AL = process number (1 to 16) // Retro UNIX 8086, 386 v1 //
                             <1>       ;      (max. 16 processes available for current kernel version)
                             <1>       ; EBX = run queue address ; 20/05/2016 (TRDOS 386)
                             <1>             ; which is one of following addresses:
                             <1>             ;  1) 'runq_event' high priority run queue
                             <1>             ;  2) 'runq_normal' normal/regular priority run queue
                             <1>             ;  3) 'runq_background' low priority run queue
                             <1>
                             <1>       ;mov   ebx, runq
0000E69D 0FB613              <1>       movzx        edx, byte [ebx]
0000E6A0 43                  <1>       inc    ebx
0000E6A1 20D2                <1>       and    dl, dl
                             <1>             ; tstb (r2)+ / is queue empty?
0000E6A3 740A                <1>             jz     short putlu_1
                             <1>             ; beq 1f / yes, branch
0000E6A5 8A13                <1>       mov    dl, [ebx] ; 12/09/2015
                             <1>             ; movb (r2),r3 / no, save the "last user" process number
                             <1>                         ; / in r3
0000E6A7 8882[9F000300]      <1>             mov    [edx+p.link-1], al
                             <1>             ; movb r1,p.link-1(r3) / put pointer to user on
                             <1>                         ; / "last users" link
0000E6AD EB03                <1>       jmp    short putlu_2
                             <1>             ; br 2f /
                             <1> putlu_1: ; 1:
0000E6AF 8843FF              <1>       mov    [ebx-1], al
                             <1>                    ; movb r1,-1(r2) / user is only user;
                             <1>                        ; / put process no. at beginning and at end
                             <1> putlu_2: ; 2:
0000E6B2 8803                <1>       mov    [ebx], al
                             <1>                    ; movb r1,(r2) / user process in r1 is now the last entry
                             <1>                         ; / on the queue
0000E6B4 88C2                <1>       mov    dl, al
0000E6B6 88B2[9F000300]      <1>         mov      [edx+p.link-1], dh ; 0
                             <1>             ; dec r2 / restore r2
0000E6BC C3                  <1>         retn
                             <1>             ; rts r0
                             <1>
                             <1> sysver:
                             <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
0000E6BD C705[64030300]0002- <1>       mov    dword [u.r0], 200h ; AH = major version, AL = minor version
0000E6C5 0000                <1>
0000E6C7 E912E0FFFF          <1>       jmp    sysret
                             <1>
                             <1>
                             <1> syspri: ; change running priority (of the process)
                             <1>       ; 21/05/2016
                             <1>       ; 20/05/2026 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; INPUT ->
                             <1>       ;     BL = priority level
                             <1>       ;        0 = low running priority (running on background)
                             <1>       ;        1 = normal/regular priority (running as regular)
                             <1>       ;        2 = high/event priority (running for event)
                             <1>       ;        >2 = invalid, it will accepted as 2 (event)
                             <1>       ;        0FFh = get/return current running priority only
                             <1>       ; OUTPUT ->
                             <1>       ;     * if current [u.pri] < 2
                             <1>       ;       if BL input < 0FFh ->
                             <1>       ;          [u.pri] is updated as in BL input (0,1,2)
                             <1>       ;       if BL input = 0FFh -> AL = [u.pri] (current)
                             <1>       ;
                             <1>       ;     * if current [u.pri] = 2
                             <1>       ;       if BL input < 0FFh -> cf = 1 & AL = 2
                             <1>       ;       if BL input = 0FFh -> cf = 0 & AL = 2
                             <1>       ;
                             <1>       ;     NOTE:
                             <1>       ;     If [u.pri] = 2, it can not be changed to 1 or 0;
                             <1>       ;     because, run queue of the running process is unspecified
                             <1>       ;     at this      stage. Process might be started by a timer event
                             <1>       ;     or priority might be changed to high by previous
                             <1>       ;     'syspri' system     call. In both cases, the process is in
                             <1>       ;     'runq_normal' or 'runq_background' queue.
                             <1>       ;     As result of this fact, when the [u.quant] time quantum
                             <1>       ;     of the process is elapsed or 'sysrele' system call is
                             <1>       ;     instructed by the process, 'tswap' ('tswitch') procedure
                             <1>       ;     will be called (to 'swap' or 'switch' out the procedure)
                             <1>       ;     and it will not call 'putlu' to add the (stopping)
                             <1>       ;     process to relevant run queue when [u.pri] = 2.
                             <1>       ;     (Otherwise, it would be possible to add process to
                             <1>       ;     a run queue while it is already in a run queue, wrongly.)
                             <1>       ;
                             <1>       ;     If [u.pri]< 2, 'tswap/tswitch' procedure will call
                             <1>       ;     'putlu' to add process to relevant run queue
                             <1>       ;     according to [u.pri] value. ('runq_normal' for 1,
                             <1>       ;     'runq_background' for 0).
                             <1>       ;
                             <1>       ;     If BL input >= 2 and < 0FFh while [u.pri] < 2,
                             <1>       ;     process will be added to 'runq_normal' queue and
                             <1>       ;     [u.pri] will be set to 2. (in 'syspri' system call)
                             <1>       ;
                             <1>
0000E6CC 29C0                <1>       sub    eax, eax ; 0
0000E6CE A3[C8030300]        <1>       mov    [u.error], eax
                             <1>
0000E6D3 A0[A9030300]        <1>       mov    al, [u.pri]
0000E6D8 A3[64030300]        <1>       mov    [u.r0], eax
                             <1>
0000E6DD FEC3                <1>       inc    bl
0000E6DF 0F84F9DFFFFF        <1>       jz     sysret ; 0FFh -> 0, get priority level
```

```
                             <1>
0000E6E5 3C02                <1>       cmp    al, 2
0000E6E7 0F83D1DFFFFF        <1>       jnb    error ; CF = 1 & AL = 2 (& last error = 0)
                             <1>
0000E6ED FECB                <1>       dec    bl
0000E6EF 80FB02              <1>       cmp    bl, 2
0000E6F2 7602                <1>       jna    short syspri_1
0000E6F4 B302                <1>       mov    bl, 2
                             <1> syspri_1:
0000E6F6 881D[A9030300]      <1>       mov    [u.pri], bl
0000E6FC 80FB02              <1>       cmp    bl, 2
0000E6FF 0F82D9DFFFFF        <1>         jb      sysret
                             <1>
                             <1>       ; here...
                             <1>       ; Priority of current process has been changed to high
                             <1>       ; ('run for event') but current process will be added to
                             <1>       ; 'run as normal' queue. ('run for event' high priority
                             <1>       ; queue is under control of timer -& RTC- interrupt only!)
                             <1>       ;
                             <1>       ; (Otherwise, process can fall into black hole!
                             <1>       ; e.g. if it is not in waiting list and it has not got
                             <1>       ; a timer event and it is not in a run queue!
                             <1>       ; Because, when [u.pri] is 2, 'tswap/tswitch' will not
                             <1>       ; add the stopping process to a run queue.)
                             <1>
0000E705 A0[B3030300]        <1>       mov    al, [u.uno]
0000E70A BB[54030300]        <1>       mov    ebx, runq_normal ; normal priority !
                             <1>                            ; [u.pri] is set to high
                             <1>                            ; but 'runq_event' queue is set
                             <1>                            ; only by the kernel's timer
                             <1>                            ; event function (timer interrupt).
0000E70F E889FFFFFF          <1>       call   putlu
0000E714 E9C5DFFFFF          <1>       jmp    sysret
                             <1>
                             <1> cpass: ; / get next character from user area of core and put it in AL (r1)
                             <1>       ; 02/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 19/05/2015 - 18/10/2015 (Retro UNIX 386 v1)
                             <1>       ; 14/08/2013 - 20/09/2013 (Retro UNIX 8086 v1)
                             <1>       ; INPUTS ->
                             <1>       ;     [u.base] = virtual address in user area
                             <1>       ;     [u.count] = byte count (max.)
                             <1>       ;     [u.pcount] = byte count in page (0 = reset)
                             <1>       ; OUTPUTS ->
                             <1>       ;     AL = the character which is pointed by [u.base]
                             <1>       ;     zf = 1 -> transfer count has been completed
                             <1>         ;
                             <1>       ; ((Modified registers:  EAX, EDX, ECX))
                             <1>       ;
0000E719 833D[88030300]00    <1>       cmp    dword [u.count], 0  ; have all the characters been transferred
                             <1>                               ; i.e., u.count, # of chars. left
0000E720 763F                <1>       jna    short cpass_3    ; to be transferred = 0?) yes, branch
0000E722 FF0D[88030300]      <1>       dec    dword [u.count]        ; no, decrement u.count
                             <1>         ; 19/05/2015
                             <1>       ;(Retro UNIX 386 v1 - translation from user's virtual address
                             <1>       ;                  to physical address
0000E728 66833D[C4030300]00  <1>       cmp    word [u.pcount], 0 ; byte count in page = 0 (initial value)
                             <1>                         ; 1-4095 --> use previous physical base address
                             <1>                         ; in [u.pbase]
0000E730 770E                <1>       ja     short cpass_1
0000E732 833D[BC030300]00    <1>       cmp     dword [u.ppgdir], 0 ; is the caller os kernel
0000E739 7427                <1>         je      short cpass_k       ; (sysexec, '/etc/init') ?  (MainProg)
0000E73B E858FDFFFF          <1>       call   trans_addr_r
                             <1> cpass_1:
0000E740 66FF0D[C4030300]    <1>       dec    word [u.pcount]
                             <1> cpass_2:
0000E747 8B15[C0030300]      <1>       mov    edx, [u.pbase]
0000E74D 8A02                <1>       mov    al, [edx]    ; take the character pointed to
                             <1>                           ; by u.base and put it in r1
0000E74F FF05[8C030300]      <1>       inc    dword [u.nread] ; increment no. of bytes transferred
0000E755 FF05[84030300]      <1>       inc    dword [u.base]  ; increment the buffer address to point to the
                             <1>                            ; next byte
0000E75B FF05[C0030300]      <1>       inc    dword [u.pbase]
                             <1> cpass_3:
0000E761 C3                  <1>       retn
                             <1> cpass_k:
                             <1>       ; 02/07/2015
                             <1>       ; The caller is os kernel
                             <1>       ; (get sysexec arguments from kernel's memory space)
0000E762 8B1D[84030300]      <1>       mov    ebx, [u.base]
0000E768 66C705[C4030300]00- <1>         mov     word [u.pcount], PAGE_SIZE ; 4096
0000E770 10                  <1>
0000E771 891D[C0030300]      <1>       mov    [u.pbase], ebx
0000E777 EBCE                <1>       jmp    short cpass_2
                             <1>
                             <1> transfer_to_user_buffer: ; fast transfer
                             <1>       ; 27/05/2016
                             <1>       ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      ESI = source address in system space
                             <1>       ;      EDI = user's buffer address
                             <1>       ;      ECX = transfer (byte) count
                             <1>       ;      [u.pgdir] = user's page directory
                             <1>       ; OUTPUT ->
                             <1>       ;      ECX = actual transfer count
                             <1>       ;      cf = 1 -> error
                             <1>       ;      [u.count] = remain byte count
                             <1>       ;
                             <1>       ; Modified registers: eax, ecx
                             <1>       ;
                             <1>
0000E779 21C9                <1>       and    ecx, ecx
0000E77B 743B                <1>       jz     short ttub_4
```

```
                                <1>
0000E77D 890D[88030300]         <1>         mov     [u.count], ecx
                                <1>
0000E783 57                     <1>         push    edi
0000E784 56                     <1>         push    esi
0000E785 53                     <1>         push    ebx
0000E786 52                     <1>         push    edx
0000E787 51                     <1>         push    ecx
                                <1>
0000E788 89FB                   <1>         mov     ebx, edi
0000E78A 81C300004000           <1>         add     ebx, CORE ; 27/05/2016
                                <1> ttub_1:
                                <1>         ; ebx = virtual (linear) address
                                <1>         ; [u.pgdir] = user's page directory
0000E790 E8006BFFFF             <1>                 call    get_physical_addr_x ; get physical address
0000E795 7222                   <1>         jc      short ttub_5
                                <1>         ; eax = physical address
                                <1>         ; ecx = remain byte count in page (1-4096)
0000E797 89C7                   <1>         mov     edi, eax
0000E799 A1[88030300]           <1>         mov     eax, [u.count]
0000E79E 39C1                   <1>         cmp     ecx, eax
0000E7A0 7602                   <1>         jna     short ttub_2
0000E7A2 89C1                   <1>         mov     ecx, eax
                                <1> ttub_2:
0000E7A4 29C8                   <1>         sub     eax, ecx
0000E7A6 01CB                   <1>         add     ebx, ecx
0000E7A8 F3A4                   <1>         rep     movsb
0000E7AA A3[88030300]           <1>         mov     [u.count], eax
0000E7AF 09C0                   <1>         or      eax, eax
0000E7B1 75DD                   <1>         jnz     short ttub_1
                                <1> ttub_retn:
                                <1> tfub_retn:
0000E7B3 59                     <1>         pop     ecx ; transfer count = actual transfer count
                                <1> ttub_3:
0000E7B4 5A                     <1>         pop     edx
0000E7B5 5B                     <1>         pop     ebx
0000E7B6 5E                     <1>         pop     esi
0000E7B7 5F                     <1>         pop     edi
                                <1> ttub_4:
0000E7B8 C3                     <1>         retn
                                <1> ttub_5:
0000E7B9 59                     <1>         pop     ecx
0000E7BA 2B0D[88030300]         <1>         sub     ecx, [u.count] ; actual transfer count
0000E7C0 F9                     <1>         stc
0000E7C1 EBF1                   <1>         jmp     short ttub_3
                                <1>
                                <1> transfer_from_user_buffer: ; fast transfer
                                <1>         ; 27/05/2016
                                <1>         ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
                                <1>         ;
                                <1>         ; INPUT ->
                                <1>         ;       ESI = user's buffer address
                                <1>         ;       EDI = destination address in system space
                                <1>         ;       ECX = transfer (byte) count
                                <1>         ;       [u.pgdir] = user's page directory
                                <1>         ; OUTPUT ->
                                <1>         ;       ecx = actual transfer count
                                <1>         ;       cf = 1 -> error
                                <1>         ;       [u.count] = remain byte count
                                <1>         ;
                                <1>         ; Modified registers: eax, ecx
                                <1>         ;
                                <1>
0000E7C3 21C9                   <1>         and     ecx, ecx
                                <1>         ;jz     short tfub_4
0000E7C5 74F1                   <1>         jz      short ttub_4
                                <1>
0000E7C7 890D[88030300]         <1>         mov     [u.count], ecx
                                <1>
0000E7CD 57                     <1>         push    edi
0000E7CE 56                     <1>         push    esi
0000E7CF 53                     <1>         push    ebx
0000E7D0 52                     <1>         push    edx
0000E7D1 51                     <1>         push    ecx
                                <1>
0000E7D2 89F3                   <1>         mov     ebx, esi
0000E7D4 81C300004000           <1>         add     ebx, CORE ; 27/05/2016
                                <1> tfub_1:
                                <1>         ; ebx = virtual (linear) address
                                <1>         ; [u.pgdir] = user's page directory
0000E7DA E8B66AFFFF             <1>                 call    get_physical_addr_x ; get physical address
                                <1>         ;jc     short tfub_5
0000E7DF 72D8                   <1>         jc      short ttub_5
                                <1>         ; eax = physical address
                                <1>         ; ecx = remain byte count in page (1-4096)
0000E7E1 89C6                   <1>         mov     esi, eax
0000E7E3 A1[88030300]           <1>         mov     eax, [u.count]
0000E7E8 39C1                   <1>         cmp     ecx, eax
0000E7EA 7602                   <1>         jna     short tfub_2
0000E7EC 89C1                   <1>         mov     ecx, eax
                                <1> tfub_2:
0000E7EE 29C8                   <1>         sub     eax, ecx
0000E7F0 01CB                   <1>         add     ebx, ecx
0000E7F2 F3A4                   <1>         rep     movsb
0000E7F4 A3[88030300]           <1>         mov     [u.count], eax
0000E7F9 09C0                   <1>         or      eax, eax
0000E7FB 75DD                   <1>         jnz     short tfub_1
                                <1>
0000E7FD EBB4                   <1>         jmp     short tfub_retn
                                <1>
                                <1> ;tfub_retn:
                                <1> ;       pop     ecx ; transfer count = actual transfer count
                                <1> ;tfub_3:
```

```
                             <1> ;     pop   edx
                             <1> ;     pop   ebx
                             <1> ;     pop   esi
                             <1> ;     pop   edi
                             <1> ;tfub_4:
                             <1> ;     retn
                             <1> ;tfub_5:
                             <1> ;     pop   ecx
                             <1> ;     sub   ecx, [u.count] ; actual transfer count
                             <1> ;     stc
                             <1> ;     jmp   short tfub_3
                             <1>
                             <1> sysfff: ; <Find First File>
                             <1>       ; 17/10/2016
                             <1>       ; 16/10/2016
                             <1>       ; 15/10/2016 TRDOS 386 (TRDOS v2.0) feature only !
                             <1>       ;           -derived from TRDOS v1.0, INT_21H.ASM-
                             <1>       ;            ("loc_INT21h_find_first_file")
                             <1>       ; TRDOS 8086 (v1.0)
                             <1>         ;   07/08/2011
                             <1>         ;   Find First File
                             <1>       ;     INPUT:
                             <1>         ;       CX= Attributes
                             <1>         ;       DS:DX= Pointer to filename
                             <1>       ;     MSDOS OUTPUT:
                             <1>       ;         DTA: (Default address: PSP offset 80h)
                             <1>       ;         Offset  Descrription
                             <1>       ;         0      Reserved for use find next file
                             <1>       ;         21     Attribute of file found
                             <1>       ;         22     Time stamp of file
                             <1>       ;         24     Date stamp of file
                             <1>       ;         26     File size in bytes
                             <1>       ;         30     Filename and extension (zero terminated)
                             <1>       ;     If cf = 1:
                             <1>       ;         Error Codes: (in AX)
                             <1>       ;            2 - File not found
                             <1>       ;            18 - No more files
                             <1>         ;
                             <1>       ; TRDOS 386 (v2.0)
                             <1>       ; 15/10/2016
                             <1>       ;
                             <1>         ; INPUT ->
                             <1>         ;      CL = File attributes
                             <1>       ;           bit 0 (1) - Read only file (R)
                             <1>       ;             bit 1 (1) - Hidden file (H)
                             <1>         ;             bit 2 (1) - System file (R)
                             <1>       ;             bit 3 (1) - Volume label/name (V)
                             <1>         ;             bit 4 (1) - Subdirectory (D)
                             <1>       ;           bit 5 (1) - File has been archived (A)
                             <1>       ;        CH = 0 -> Return basic parameters (24 bytes)
                             <1>       ;        CH > 0 -> Return FindFile structure/table (128 bytes)
                             <1>         ;          EBX = Pointer to filename (ASCIIZ) -path-
                             <1>       ;        EDX = File parameters buffer address
                             <1>       ;            (buffer size = 24 bytes if CH input = 0)
                             <1>       ;            (buffer size = 128 bytes if CH input > 0)
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;        EAX = 0 if CH input > 0
                             <1>       ;        EAX = First cluster number of file if CH input = 0
                             <1>       ;        EDX = File parameters table/structure address
                             <1>       ;        Basic Parameters:
                             <1>       ;            Offset  Description
                             <1>       ;            ------ ---------------
                             <1>       ;            0      File Attributes
                             <1>       ;            1      Ambiguous filename chars are used sign
                             <1>       ;                   (0 = filename fits exactly with request)
                             <1>       ;                   (>0 = ambiguous filename chars are used)
                             <1>       ;            2      Time stamp of file
                             <1>       ;            4      Date stamp of file
                             <1>       ;            6      File size in bytes
                             <1>       ;            10     Short Filename (ASCIIZ, max. 13 bytes)
                             <1>       ;            23     Longname Length (1-255) if existing
                             <1>       ;
                             <1>       ;          cf = 1 -> Error code in AL
                             <1>       ;
                             <1>       ; Modified Registers: EAX (at the return of system call)
                             <1>       ;
                             <1>       ; TR-DOS FindFile (FFF) Structure (128 bytes):
                             <1>       ; 09/10/2011 (DIR.ASM) - 10/02/2016 (trdoskx.s)
                             <1>       ;
                             <1>       ; Offset     Parameter           Size
                             <1>       ; ------     ------------------  --------
                             <1>       ; 0          FindFile_Drv        1 byte
                             <1>       ; 1          FindFile_Directory  65 bytes
                             <1>       ; 66         FindFile_Name       13 bytes
                             <1>       ; 79         FindFile_LongNameEntryLength 1 byte
                             <1>       ;Above 80 bytes form
                             <1>       ;TR-DOS Source/Destination File FullName Format/Structure
                             <1>       ; 80         FindFile_AttributesMask 1 word
                             <1>       ; 82         FindFile_DirEntry   32 bytes (*)
                             <1>       ; 114        FindFile_DirFirstCluster 1 double word
                             <1>       ; 118        FindFile_DirCluster 1 double word
                             <1>       ; 122        FindFile_DirEntryNumber 1 word
                             <1>       ; 124        FindFile_MatchCounter     1 word
                             <1>       ; 126        FindFile_Reserved   1 word
                             <1>       ; (*) MS-DOS, FAT 12-16-32 classic directory entry (32 bytes)
                             <1>
                             <1>       ;mov   [u.namep], ebx
                             <1>       ; 16/10/2016
0000E7FF 8915[EC650100]      <1>       mov    [FFF_UBuffer], edx
0000E805 66890D[F1650100]    <1>       mov    [FFF_Attrib], cx ; [FFF_RType] = ch
                             <1>                ; Attributes in CL, return data type in CH
0000E80C 89DE                <1>       mov    esi, ebx
```

```
                              <1>         ; file name is forced, change directory as temporary
                              <1>         ;mov   ax, 1
                              <1>         ;mov   [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
                              <1>         ;call  set_working_path
0000E80E E8E8130000           <1>         call   set_working_path_x ; 17/10/2016
0000E813 731D                 <1>         jnc    short sysfff_0
                              <1>
0000E815 21C0                 <1>         and    eax, eax  ; 0 -> Bad Path!
0000E817 7505                 <1>         jnz    short sysfff_err
                              <1>
                              <1>         ; eax = 0
0000E819 B80C000000           <1>         mov    eax, ERR_DIR_NOT_FOUND ; Directory not found !
                              <1> sysfff_err:
0000E81E A3[64030300]         <1>         mov    [u.r0], eax
0000E823 A3[C8030300]         <1>         mov    [u.error], eax
0000E828 E8A3140000           <1>           call       reset_working_path
0000E82D E98CDEFFFF           <1>         jmp    error
                              <1>
                              <1> sysfff_0:
                              <1>         ;sub   ah, ah ; ah = 0
0000E832 8A0424               <1>         mov    al, [esp]
0000E835 08C0                 <1>         or     al, al
0000E837 7412                 <1>         jz     short sysfff_2
0000E839 B410                 <1>         mov    ah, 10h
0000E83B A808                 <1>         test   al, 08h
0000E83D 7503                 <1>         jnz    short sysfff_1
0000E83F 80CC08               <1>         or     ah, 08h
                              <1> sysfff_1:
0000E842 2410                 <1>         and    al, 10h ; Directory
0000E844 7405                 <1>         jz     short sysfff_2
0000E846 80E408               <1>         and    ah, 08h
0000E849 30C0                 <1>         xor    al, al ; When a directory is searched,
                              <1>                       ; filename will be returned even if
                              <1>                       ; it is not a directory!
                              <1>                       ; Because: (in order to prevent
                              <1>                       ; creating a dir with existing file name)
                              <1>                       ; Dir and file names must not be same!
                              <1>                       ; (return attribute must be checked)
                              <1> sysfff_2:
                              <1>         ; AX = Attributes mask
                              <1>                ; AL = AND mask (result must be equal to AL)
                              <1>                ; AH = Negative AND mask (result must be ZERO)
                              <1>         ; ESI = FindFile_Name address
                              <1>
0000E84B E8139AFFFF           <1>         call   find_first_file
0000E850 72CC                 <1>         jc     short sysfff_err ; eax = 2 (File not found !)
                              <1>
                              <1>         ; ESI = Directory Entry (FindFile_DirEntry) Location
                              <1>         ; EDI = Directory Buffer Directory Entry Location
                              <1>         ; EAX = File Size
                              <1>         ;  BL = Attributes of The File/Directory
                              <1>         ;  BH = Long Name Yes/No Status (>0 is YES)
                              <1>         ;  DX > 0 : Ambiguous filename chars are used
                              <1>
                              <1> sysfff_3:
                              <1>         ; 16/10/2016
0000E852 668B0D[F1650100]     <1>         mov    cx, [FFF_Attrib]
                              <1>         ; Attribs in CL, return data type in CH
                              <1>
                              <1>         ;or    cl, cl
                              <1>         ;jz    short sysfff_4 ; 0 = No filter
0000E859 80F1FF               <1>         xor    cl, 0FFh
0000E85C 20D9                 <1>         and    cl, bl
0000E85E 7409                 <1>         jz     short sysfff_4
                              <1>
                              <1>         ;mov   eax, 2 ; 'file not found !' error
                              <1>         ;jmp   short sysfff_err_1
                              <1>
                              <1>         ; 16/10/2016
0000E860 E8AD9AFFFF           <1>         call   find_next_file
0000E865 72B7                 <1>         jc     short sysfff_err ; eax = 12 (no more files !)
0000E867 EBE9                 <1>         jmp    short sysfff_3
                              <1>
                              <1> sysfff_4:
0000E869 20ED                 <1>         and    ch, ch ; [FFF_RType]
0000E86B 7412                 <1>         jz     short sysfff_5
0000E86D B980000000           <1>         mov    ecx, 128 ; ; transfer length
0000E872 880D[F0650100]       <1>         mov    [FFF_Valid], cl
                              <1> sysfnf_11:
0000E878 BE[A2620100]         <1>         mov    esi, FindFile_Drv
0000E87D EB44                 <1>         jmp    short sysfff_6
                              <1> sysfff_5:
                              <1>         ;mov   esi, FindFile_DirEntry
0000E87F B918000000           <1>         mov    ecx, 24  ; transfer length
0000E884 880D[F0650100]       <1>         mov    [FFF_Valid], cl
                              <1> sysfnf_12:
0000E88A BF[AC6A0100]         <1>         mov    edi, DTA ; FFF data transfer address
                              <1>         ;mov   al, [esi+DirEntry_Attr] ; 11
0000E88F 88D8                 <1>         mov    al, bl ; File/Dir Attributes
0000E891 887F17               <1>         mov    [edi+23], bh ; Longname length (0= none)
0000E894 AA                   <1>         stosb
0000E895 88D0                 <1>         mov    al, dl ; DL is for '?'
0000E897 00F0                 <1>         add    al, dh ; DH is for '*'
                              <1>         ; AL > 0 if ambiguous file name wildcards are used
0000E899 AA                   <1>         stosb
0000E89A 8B4616               <1>         mov    eax, [esi+DirEntry_WrtTime] ; 22
0000E89D AB                   <1>           stosd      ; DirEntry_WrtTime & DirEntry_WrtDate
0000E89E 8B461C               <1>           mov eax, [esi+DirEntry_FileSize] ; 28
0000E8A1 AB                   <1>           stosd
0000E8A2 668B4614             <1>         mov    ax, [esi+DirEntry_FstClusHI] ; 20
0000E8A6 66C1E010             <1>         shl    ax, 16
0000E8AA 668B461A             <1>         mov    ax, [esi+DirEntry_FstClusLO] ; 26
0000E8AE A3[64030300]         <1>         mov    [u.r0], eax ; First Cluster
```

```
                             <1> 
                             <1>         ;mov esi, FindFile_DirEntry
0000E8B3 E855140000          <1>       call   get_file_name
                             <1> 
0000E8B8 8A0D[F0650100]      <1>       mov    cl, [FFF_Valid]
0000E8BE BE[AC6A0100]        <1>              mov    esi, DTA ; FFF data transfer address
                             <1> sysfff_6:
0000E8C3 8B3D[EC650100]      <1>       mov    edi, [FFF_UBuffer] ; user's buffer address (edx)
0000E8C9 E8ABFEFFFF          <1>       call   transfer_to_user_buffer
                             <1> 
0000E8CE 890D[64030300]      <1>       mov    [u.r0], ecx ; actual transfer count
0000E8D4 E8F7130000          <1>         call       reset_working_path
0000E8D9 E900DEFFFF          <1>       jmp    sysret
                             <1> 
                             <1> sysfnf: ; <Find Next File>
                             <1>       ; 16/10/2016 TRDOS 386 (TRDOS v2.0) feature only !
                             <1>       ;           -derived from TRDOS v1.0, INT_21H.ASM-
                             <1>       ;            ("loc_INT21h_find_next_file")
                             <1>       ; TRDOS 8086 (v1.0)
                             <1>         ;    07/08/2011
                             <1>         ;    Find First File
                             <1>       ;     INPUT:
                             <1>         ;       none
                             <1>       ;     MSDOS OUTPUT:
                             <1>       ;         DTA: (Default address: PSP offset 80h)
                             <1>       ;         Offset  Descrription
                             <1>       ;         0      Reserved for use find next file
                             <1>       ;         21     Attribute of file found
                             <1>       ;         22     Time stamp of file
                             <1>       ;         24     Date stamp of file
                             <1>       ;         26     File size in bytes
                             <1>       ;         30     Filename and extension (zero terminated)
                             <1>       ;     If cf = 1:
                             <1>       ;         Error Codes: (in AX)
                             <1>       ;            18 - No more files
                             <1>         ;
                             <1>       ; TRDOS 386 (v2.0)
                             <1>       ; 16/10/2016
                             <1>       ;
                             <1>         ; INPUT ->
                             <1>              ;         none
                             <1>       ; OUTPUT ->
                             <1>       ;        EAX = 0 if CH input of 'Find First File' > 0
                             <1>       ;        EAX = First cluster number of file
                             <1>       ;             if CH input of 'Find First File' = 0
                             <1>       ;        EDX = File parameters table/structure address
                             <1>       ;
                             <1>       ;         cf = 1 -> Error code in AL
                             <1>       ;
                             <1>       ; Modified Registers: EAX (at the return of system call)
                             <1> 
                             <1>       ;
                             <1>       ; Note: If byte [FFF_Valid] = 0
                             <1>       ;     'sysfnf' will return with 'no more files' error.
                             <1>       ;     If byte [FFF_Valid] = 24
                             <1>       ;     'sysfnf' will return with 32 bytes basic parameters
                             <1>       ;     at the address which is in EDX.
                             <1>       ;     If byte [FFF_Valid] = 128
                             <1>       ;     'sysfnf' will return with 128 bytes Find File
                             <1>       ;     Structure/Table at the address which is in EDX.
                             <1> 
0000E8DE 803D[F0650100]00    <1>       cmp    byte [FFF_Valid], 0
0000E8E5 7714                <1>       ja     short stsfnf_0
                             <1>       ; 'no more files !' error
0000E8E7 B80C000000          <1>       mov    eax, ERR_NO_MORE_FILES ; 12
0000E8EC A3[64030300]        <1>       mov    [u.r0], eax
0000E8F1 A3[C8030300]        <1>       mov    [u.error], eax
0000E8F6 E9C3DDFFFF          <1>       jmp    error
                             <1> stsfnf_0:
                             <1>       ;cmp   byte [FFF_Valid], 128
                             <1>       ;je    short stsfnf_1
                             <1>       ;cmp   byte [FFF_Valid], 24
                             <1>       ;je    short stsfnf_1
                             <1>       ;mov   [FFF_Valid], 24 ; Default
                             <1> stsfnf_1:
0000E8FB 0FB61D[FE580100]    <1>       movzx  ebx, byte [Current_Drv]
0000E902 66891D[F6650100]    <1>       mov    [SWP_DRV], bx
0000E909 8A15[A2620100]      <1>       mov    dl, [FindFile_Drv]
0000E90F 38DA                <1>       cmp    dl, bl
0000E911 750B                <1>       jne    short stsfnf_2
0000E913 86FB                <1>       xchg   bh, bl
0000E915 BE00010900          <1>       mov    esi, Logical_DOSDisks
0000E91A 01DE                <1>       add    esi, ebx
0000E91C EB0D                <1>       jmp    short sysfnf_3
                             <1> 
                             <1> stsfnf_2:
0000E91E FE05[F7650100]      <1>       inc    byte [SWP_DRV_chg]
                             <1> 
0000E924 E89785FFFF          <1>       call   change_current_drive
0000E929 7245                <1>       jc     short sysfnf_err_1 ; read error !
                             <1>                              ; (do not stop, because
                             <1>                              ; we don't have a
                             <1>                              ; 'no more files'
                             <1>                              ; -file not found- error,
                             <1>                              ; next sysfnf system call
                             <1>                              ; may solve the problem,
                             <1>                              ; after re-placing the disk)
                             <1> sysfnf_3:
0000E92B A1[18630100]        <1>       mov    eax, [FindFile_DirCluster]
0000E930 21C0                <1>       and    eax, eax
0000E932 7550                <1>       jnz    short sysfnf_6
                             <1> 
0000E934 803D[FD580100]02    <1>       cmp    byte [Current_FATType], 2
```

```
0000E93B 772C                <1>       ja     short sysfnf_err_0 ; invalid, we neeed to stop !?
0000E93D 803D[FD580100]01    <1>       cmp    byte [Current_FATType], 1
0000E944 7223                <1>       jb     short sysfnf_err_0 ; invalid, we neeed to stop !?
                             <1>
0000E946 3805[28610100]      <1>       cmp    byte [DirBuff_ValidData], al ; 0
0000E94C 7608                <1>       jna    short sysfnf_4
                             <1>
0000E94E 3B05[2D610100]      <1>       cmp    eax, [DirBuff_Cluster] ; 0 ?
0000E954 745E                <1>       je     short sysfnf_9
                             <1>
                             <1>       ;cmp   byte [Current_Dir_Level], 0
                             <1>         ;ja  short sysfnf_4
                             <1>         ;jna short sysfnf_9
                             <1>
                             <1> sysfnf_4:
0000E956 FE05[F7650100]      <1>       inc    byte [SWP_DRV_chg]
0000E95C E842D3FFFF          <1>       call   load_FAT_root_directory
0000E961 7351                <1>       jnc    short sysfnf_9
                             <1>       ; eax = error code (17, 'drv not ready or read error')
0000E963 EB0B                <1>       jmp    short sysfnf_err_1 ; read error ! (no FNF stop)
                             <1>                              ; (if you want, try again,
                             <1>                              ;  after re-placing the disk)
                             <1> sysfnf_5:
0000E965 3C0C                <1>       cmp    al, 12 ; 'no more files' error
0000E967 7507                <1>       jne    short sysfnf_err_1 ; (no FNF stop -sysfnf will try
                             <1>                              ;  to read the directory again,
                             <1>                              ;  if the user calls sysfnf
                             <1>                              ;  just after this error return-)
                             <1>       ; (FNF stop -sysfnf will not try
                             <1>       ;  to read the directory again-)
                             <1>
                             <1> sysfnf_err_0:
0000E969 C605[F0650100]00    <1>       mov    byte [FFF_Valid], 0 ; FNF stop sign
                             <1> sysfnf_err_1:
0000E970 A3[64030300]        <1>       mov    [u.r0], eax
0000E975 A3[C8030300]        <1>       mov    [u.error], eax
0000E97A E851130000          <1>       call   reset_working_path
0000E97F E93ADDFFFF          <1>       jmp    error
                             <1>
                             <1> sysfnf_6:
0000E984 803D[28610100]00    <1>       cmp    byte [DirBuff_ValidData], 0
0000E98B 7608                <1>       jna    short sysfnf_7
                             <1>
0000E98D 3B05[2D610100]      <1>       cmp    eax, [DirBuff_Cluster]
0000E993 741F                <1>       je     short sysfnf_9
                             <1>
                             <1> sysfnf_7:
0000E995 FE05[F7650100]      <1>       inc    byte [SWP_DRV_chg]
0000E99B 803D[FD580100]01    <1>       cmp    byte [Current_FATType], 1
0000E9A2 7309                <1>       jnb    short sysfnf_8
                             <1>
                             <1>       ; Singlix (TRFS) File System
                             <1>       ; (access via compatibility buffer)
0000E9A4 E8C2D3FFFF          <1>       call   load_FS_sub_directory
0000E9A9 7309                <1>       jnc    short sysfnf_9
                             <1>
0000E9AB EBC3                <1>       jmp    short sysfnf_err_1 ; read error (no FNF stop)
                             <1>
                             <1> sysfnf_8:
0000E9AD E87CD3FFFF          <1>       call   load_FAT_sub_directory
0000E9B2 72BC                <1>       jc     short sysfnf_err_1 ; read error (no FNF stop)
                             <1>
                             <1> sysfnf_9:
0000E9B4 E85999FFFF          <1>       call   find_next_file
0000E9B9 72AA                <1>       jc     short sysfnf_5
                             <1>
0000E9BB A0[F1650100]        <1>       mov    al, [FFF_Attrib]
                             <1>       ;or    al, al
                             <1>       ;jz    short sysfnf_10 ; 0 = No filter
0000E9C0 34FF                <1>       xor    al, 0FFh
0000E9C2 20D8                <1>       and    al, bl
0000E9C4 75EE                <1>       jnz    short sysfnf_9 ; search for next file until
                             <1>                           ; an error return from
                             <1>                           ; find_next_file procedure
                             <1> sysfnf_10:
0000E9C6 0FB60D[F0650100]    <1>         movzx      ecx, byte [FFF_Valid]
0000E9CD 80F980              <1>       cmp    cl, 128 ; complete FindFile structure/table
0000E9D0 0F84A2FEFFFF        <1>       je     sysfnf_11
                             <1>       ;cmp   cl, 24  ; basic parameters
                             <1>       ;je    sysfnf_12
0000E9D6 E9AFFEFFFF          <1>       jmp    sysfnf_12
                             <1>
                             <1> writei:
                             <1>       ; 26/10/2016
                             <1>       ; 25/10/2016
                             <1>       ; 23/10/2016
                             <1>       ; 22/10/2016
                             <1>       ; 19/10/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 19/05/2015 - 20/05/2015 (Retro UNIX 386 v1)
                             <1>       ; 12/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; Write data to file with first cluster number in EAX
                             <1>       ;
                             <1>       ; INPUTS ->
                             <1>       ;    EAX - First cluster number of the file
                             <1>       ;    EBX - File number (Open file index number)
                             <1>       ;    u.count - byte count to be written
                             <1>       ;    u.base - points to user buffer
                             <1>       ;    u.fofp - points to dword with current file offset
                             <1>       ;    i.size - file size
                             <1>       ;    cdev - logical dos drive number of the file
                             <1>       ; OUTPUTS ->
                             <1>       ;    u.count - cleared
                             <1>       ;    u.nread - accumulates total bytes passed back
                             <1>       ;    i.size - new file size (if file byte offset overs file size)
                             <1>       ;    u.fofp - points to u.off (with new offset value)
                             <1>       ;
                             <1>       ; (Retro UNIX Prototype : 11/11/2012 - 18/11/2012, UNIXCOPY.ASM)
                             <1>       ; ((Modified registers: eax, edx, ebx, ecx, esi, edi, ebp))
                             <1>
0000E9DB 31C9                <1>       xor    ecx, ecx
0000E9DD 890D[8C030300]      <1>       mov    [u.nread], ecx  ; 0
0000E9E3 66890D[C4030300]    <1>       mov    [u.pcount], cx ; 19/05/2015
0000E9EA 390D[88030300]      <1>       cmp    [u.count], ecx
0000E9F0 7701                <1>       ja     short writei_1
0000E9F2 C3                  <1>       retn
                             <1> writei_1:
0000E9F3 881D[B0650100]      <1>       mov    [writei.ofn], bl ; Open file number
0000E9F9 880D[EB650100]      <1>       mov    [setfmod], cl ; 0 ; reset 'update lm date&time' sign
                             <1> dskw_0:
                             <1>       ; 26/10/2016
                             <1>       ; 22/10/2016, 23/10/2016, 25/10/2016
                             <1>       ; 19/10/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; 31/05/2015 - 25/07/2015 (Retro UNIX 386 v1)
                             <1>       ; 26/04/2013 - 20/09/2013 (Retro UNIX 8086 v1)
                             <1>       ;
                             <1>       ; 01/08/2013 (mkdir_w check)
0000E9FF E8D7000000          <1>       call   mget_w
                             <1>       ; eax = sector/block number
                             <1>
0000EA04 8B1D[74030300]      <1>       mov     ebx, [u.fofp]
0000EA0A 8B13                <1>       mov    edx, [ebx]
0000EA0C 81E2FF010000        <1>       and    edx, 1FFh  ; / test the lower 9 bits of the file offset
0000EA12 750C                <1>       jnz    short dskw_1 ; / if its non-zero, branch
                             <1>                         ; if zero, file offset = 0,
                             <1>                               ; / 512, 1024,...(i.e., start of new block)
0000EA14 813D[88030300]0002- <1>       cmp    dword [u.count], 512
0000EA1C 0000                <1>
                             <1>                           ; / if zero, is there enough data to fill
                             <1>                           ; / an entire block? (i.e., no. of
0000EA1E 7337                <1>       jnb    short dskw_2 ; / bytes to be written greater than 512.?
                             <1>                         ; / Yes, branch. Don't have to read block
                             <1> dskw_1: ; in as no past info. is to be saved
                             <1>       ; (the entire block will be overwritten).
                             <1>       ; 23/10/2016
                             <1>
0000EA20 BB[94070300]        <1>       mov    ebx, writei_buffer
                             <1>       ; esi = logical dos drive description table address
                             <1>       ; eax = sector number
                             <1>       ; ebx = buffer address (in kernel's memory space)
                             <1>       ; ecx = sector count
0000EA25 B901000000          <1>       mov    ecx, 1
0000EA2A E8A90D0000          <1>       call   disk_read
                             <1>       ;call  dskrd  ; / no, must retain old info..
                             <1>                           ; / Hence, read block 'r1' into an I/O buffer
0000EA2F 7326                <1>       jnc    short dskw_2
                             <1>
                             <1>       ; disk read error
0000EA31 B811000000          <1>       mov    eax, 17 ; drive not ready or READ ERROR !
                             <1> dskw_err: ; jump from disk write error
0000EA36 A3[64030300]        <1>       mov    [u.r0], eax
0000EA3B A3[C8030300]        <1>       mov    [u.error], eax
                             <1>
0000EA40 803D[EB650100]00    <1>       cmp    byte [setfmod], 0
0000EA47 0F8671DCFFFF        <1>       jna    error
                             <1>
0000EA4D E8AF030000          <1>       call   update_file_lmdt ; update last modif. date&time of the file
                             <1>       ;mov   byte [setfmod], 0
                             <1>
0000EA52 E967DCFFFF          <1>       jmp    error
                             <1>
                             <1> dskw_2: ; 3:
                             <1>       ; 23/10/2016
0000EA57 C605[8C650100]01    <1>       mov    byte [writei.valid], 1 ; writei buffer contains valid data
0000EA5E 56                  <1>       push   esi ; logical dos drive description table address
                             <1>       ; EAX (r1) = block/sector number
                             <1>       ;call  wslot
                             <1>              ; jsr r0,wslot / set write and inhibit bits in I/O queue,
                             <1>                       ; / proc. status=0, r5 points to 1st word of data
0000EA5F 803D[C6030300]00    <1>       cmp    byte [u.kcall], 0
0000EA66 770F                <1>       ja     short dskw_4 ; zf=0 -> the caller is 'mkdir'
                             <1>       ;
0000EA68 66833D[C4030300]00  <1>       cmp    word [u.pcount], 0
0000EA70 7705                <1>       ja     short dskw_4
                             <1> dskw_3:
                             <1>       ; [u.base] = virtual address to transfer (as source address)
0000EA72 E821FAFFFF          <1>       call   trans_addr_r ; translate virtual address to physical (r)
                             <1> dskw_4:
0000EA77 BB[94070300]        <1>       mov    ebx, writei_buffer
                             <1>       ; EBX (r5) = system (I/O) buffer address
0000EA7C E883FAFFFF          <1>       call   sioreg
                             <1>       ; ESI = file (user data) offset
                             <1>       ; EDI = sector (I/O) buffer offset
                             <1>       ; ECX = byte count
                             <1>       ;
0000EA81 F3A4                <1>       rep    movsb
                             <1>       ; 25/07/2015
                             <1>       ; eax = remain bytes in buffer
                             <1>         ;       (check if remain bytes in the buffer > [u.pcount])
0000EA83 09C0                <1>       or     eax, eax
0000EA85 75EB                <1>       jnz    short dskw_3 ; (page end before system buffer end!)
                             <1>
                             <1>       ; 23/10/2016
0000EA87 B101                <1>       mov    cl, 1
0000EA89 5E                  <1>       pop    esi
0000EA8A A1[90650100]        <1>       mov    eax, [writei.sector]
```

```
                            <1>         ; esi = logical dos drive description table address
                            <1>         ; eax = sector number
                            <1>         ; ebx = writei buffer address
                            <1>         ; ecx = sector count
0000EA8F E8350D0000         <1>         call   disk_write ; / yes, write the block
0000EA94 7307               <1>         jnc    short dskw_5
                            <1>
0000EA96 B812000000         <1>         mov    eax, 18 ; drive not ready or WRITE ERROR !
0000EA9B EB99               <1>         jmp    short dskw_err
                            <1>
                            <1> dskw_5:
                            <1>         ; 26/10/2016
0000EA9D 0FB61D[B0650100]   <1>         movzx  ebx, byte [writei.ofn] ; open file number
0000EAA4 C0E302             <1>         shl    bl, 2 ; *4
0000EAA7 8B83[80690100]     <1>         mov    eax, [ebx+OF_POINTER]
0000EAAD 3B83[A8690100]     <1>         cmp    eax, [ebx+OF_SIZE]
0000EAB3 7606               <1>         jna    short dskw_6
0000EAB5 8983[A8690100]     <1>         mov    [ebx+OF_SIZE], eax
                            <1> dskw_6:
                            <1>         ;shr   bl, 2
0000EABB 833D[88030300]00   <1>           cmp     dword [u.count], 0 ; / any more data to write?
0000EAC2 760A               <1>         jna    short dskw_7
0000EAC4 A1[A0650100]       <1>         mov    eax, [writei.fclust]
0000EAC9 E931FFFFFF         <1>         jmp    dskw_0 ; / yes, branch
                            <1> dskw_7:
                            <1>         ; update last modif. date&time of the file
                            <1>         ; (also updates file size as OF_SIZE)
0000EACE E82E030000         <1>         call   update_file_lmdt
                            <1>         ;mov   byte [setfmod], 0
                            <1>
                            <1>         ; 03/08/2013
0000EAD3 C605[C6030300]00   <1>         mov    byte [u.kcall], 0
                            <1>         ; 23/10/2016
                            <1>         ;mov   eax, [writei.fclust]
0000EADA C3                 <1>         retn
                            <1>
                            <1> mget_w:
                            <1>         ; 02/11/2016
                            <1>         ; 01/11/2016
                            <1>         ; 23/10/2016, 31/10/2016
                            <1>         ; 22/10/2016 - TRDOS 386 (TRDOS v2.0)
                            <1>         ; 03/06/2015 (Retro UNIX 386 v1, 'mget', u.5s)
                            <1>         ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
                            <1>         ;
                            <1>         ; Get existing or (allocate) a new disk block for file
                            <1>         ;
                            <1>         ; INPUTS ->
                            <1>         ;    [u.fofp] = file offset pointer
                            <1>         ;    [i.size] = file size
                            <1>         ;    [u.count] = byte count
                            <1>         ;    EAX = First cluster
                            <1>         ;    [cdev] = Logical dos drive number
                            <1>         ;    [writei.ofn] = File Number
                            <1>         ;                (Open file index, 0 based)
                            <1>         ;    ([u.off] = file offset)
                            <1>         ; OUTPUTS ->
                            <1>         ;    EAX = logical sector number
                            <1>         ;    ESI = Logical Dos Drive Description Table address
                            <1>         ;
                            <1>         ; Modified registers: EDX, EBX, ECX, ESI, EDI, EBP
                            <1>
0000EADB 8B35[74030300]     <1>           mov     esi, [u.fofp]
0000EAE1 8B2E               <1>         mov    ebp, [esi] ; u.off (or EBX*4+OF_POINTER)
                            <1>
0000EAE3 29C9               <1>         sub    ecx, ecx
0000EAE5 8A2D[46030300]     <1>         mov    ch, [cdev]
                            <1>
0000EAEB BE00010900         <1>         mov    esi, Logical_DOSDisks
0000EAF0 01CE               <1>         add    esi, ecx
                            <1>
                            <1>         ; 31/10/2016
0000EAF2 89C3               <1>         mov    ebx, eax ; First Cluster or FDT address
                            <1>
0000EAF4 807E0300           <1>         cmp    byte [esi+LD_FATType], 0
0000EAF8 0F86DD010000       <1>         jna    mget_w_14 ; Singlix FS
                            <1>
0000EAFE 0FB74611           <1>         movzx  eax, word [esi+LD_BPB+BytesPerSec]
0000EB02 0FB65613           <1>         movzx  edx, byte [esi+LD_BPB+SecPerClust]
0000EB06 8815[8E650100]     <1>         mov    [writei.spc], dl  ; sectors per cluster
0000EB0C F7E2               <1>         mul    edx
                            <1>         ; edx = 0
                            <1>         ; eax = bytes per cluster (<= 65536)
                            <1>
                            <1>         ; 02/11/2016
0000EB0E 89C1               <1>         mov    ecx, eax
0000EB10 48                 <1>         dec    eax
0000EB11 66A3[94650100]     <1>         mov    [writei.bpc], ax
                            <1>
0000EB17 89E8               <1>         mov    eax, ebp
0000EB19 0305[88030300]     <1>         add    eax, [u.count] ; next file position
0000EB1F 3B05[55040300]     <1>         cmp    eax, [i.size] ; <= file size ?
0000EB25 0F86FC000000       <1>         jna    mget_w_4 ; no
                            <1>
0000EB2B F7F1               <1>         div    ecx
0000EB2D A3[9C650100]       <1>         mov    [writei.c_index], eax ; cluster index
                            <1>         ; edx = byte offset in cluster (<= 65535)
                            <1>         ;mov   [writei.offset], dx
                            <1>         ;shr   dx, 9 ; / 512
                            <1>         ;mov   [writei.s_index], dl ; sector index in cluster (0 to spc -1)
                            <1>
0000EB32 29D2               <1>         sub    edx, edx ; 01/11/2016
0000EB34 8915[90650100]     <1>         mov    [writei.sector], edx ; 0
0000EB3A 668915[96650100]   <1>         mov    [writei.offset], dx  ; byte offset in cluster
```

```
0000EB41 8815[8F650100]         <1>         mov    [writei.s_index], dl ; sector index in cluster (0 to spc -1)
                                <1> 
0000EB47 89D8                   <1>         mov    eax, ebx ; First Cluster
                                <1> 
                                <1>         ; is this the 1st mget_w or a next mget_w call ? (by 'writei')
0000EB49 3815[8C650100]         <1>         cmp    byte [writei.valid], dl ; 0
0000EB4F 7624                   <1>         jna    short mget_w_0
                                <1> 
0000EB51 8815[8C650100]         <1>         mov    byte [writei.valid], dl ; 0 ; reset ('writei' will set it)
                                <1> 
0000EB57 3B05[A0650100]         <1>         cmp    eax, [writei.fclust]
0000EB5D 7516                   <1>         jne    short mget_w_0
                                <1> 
0000EB5F 8A0D[46030300]         <1>         mov    cl, [cdev]
0000EB65 3A0D[8D650100]         <1>         cmp    cl, [writei.drv]
0000EB6B 7508                   <1>         jne    short mget_w_0
                                <1>         ; [writei.l_clust] & [writei.l_index] are valid,
                                <1>         ;  we don't need to get last cluster & last cluster index
0000EB6D 8B0D[AC650100]         <1>         mov    ecx, [writei.l_index]
0000EB73 EB64                   <1>         jmp    short mget_w_2
                                <1> mget_w_0:
0000EB75 A3[A0650100]           <1>         mov    [writei.fclust], eax ; first cluster
                                <1>         ; edx = 0
0000EB7A A3[98650100]           <1>         mov    [writei.cluster], eax ; first cluster ; 01/11/2016
0000EB7F 8915[A4650100]         <1>         mov    [writei.fs_index], edx ; 0 ; curret cluster index
                                <1> 
                                <1>         ; FAT file system (FAT12, FAT16, FAT32)
0000EB85 E8B9D7FFFF             <1>         call   get_last_cluster
0000EB8A 0F822B010000           <1>         jc     mget_w_err ; eax = error code
                                <1> 
0000EB90 A3[A8650100]           <1>         mov    [writei.lclust], eax ; last cluster
                                <1> 
0000EB95 8B0D[CC630100]         <1>         mov    ecx, [glc_index] ; last cluster index
0000EB9B 890D[AC650100]         <1>         mov    [writei.l_index], ecx
                                <1> 
0000EBA1 A0[B0650100]           <1>         mov    al, [writei.ofn]
0000EBA6 FEC0                   <1>         inc    al
0000EBA8 A2[EB650100]           <1>         mov    [setfmod], al ; update lm date&time sign
                                <1> 
                                <1> mget_w_1:
0000EBAD 3B0D[9C650100]         <1>         cmp    ecx, [writei.c_index]  ; last cluster index
0000EBB3 7324                   <1>         jnb    short mget_w_2 ; 01/11/2016
                                <1> 
0000EBB5 A1[A8650100]           <1>         mov    eax, [writei.lclust]
                                <1>         ; EAX = Last cluster
0000EBBA E892D8FFFF             <1>         call   add_new_cluster
0000EBBF 0F82F6000000           <1>         jc     mget_w_err ; eax = error code
                                <1>         ; edx = 0
0000EBC5 A3[A8650100]           <1>         mov    [writei.lclust], eax ; (new) last cluster
0000EBCA 8B0D[AC650100]         <1>         mov    ecx, [writei.l_index]
0000EBD0 41                     <1>         inc    ecx ; add 1 to last cluster index
0000EBD1 890D[AC650100]         <1>         mov    [writei.l_index], ecx ; current last cluster index
                                <1> 
0000EBD7 EBD4                   <1>         jmp    short mget_w_1
                                <1> 
                                <1> mget_w_2:
0000EBD9 89E9                   <1>         mov    ecx, ebp
0000EBDB 030D[88030300]         <1>         add    ecx, [u.count]
0000EBE1 890D[55040300]         <1>         mov    [i.size], ecx ; save new file size
                                <1>         ;sub   edx, edx ; 0
                                <1> 
0000EBE7 A0[46030300]           <1>         mov    al, [cdev]
0000EBEC A2[8D650100]           <1>         mov    [writei.drv], al ; physical drive number
                                <1>         ; edx = 0
0000EBF1 89E8                   <1>         mov    eax, ebp ; file offset
0000EBF3 0FB70D[94650100]       <1>         movzx  ecx, word [writei.bpc] ; bytes per cluster - 1
0000EBFA 41                     <1>         inc    ecx ; bytes per cluster
0000EBFB F7F1                   <1>         div    ecx
                                <1>         ; edx = byte offset in cluster (<= 65535)
                                <1>         ; eax = cluster index
0000EBFD A3[9C650100]           <1>         mov    [writei.c_index], eax
0000EC02 668915[96650100]       <1>         mov    [writei.offset], dx
0000EC09 66C1EA09               <1>         shr    dx, 9 ; / 512
0000EC0D 8815[8F650100]         <1>         mov    [writei.s_index], dl ; sector index in cluster (0 to spc -1)
                                <1> 
                                <1> mget_w_3:
0000EC13 3B05[AC650100]         <1>         cmp    eax, [writei.l_index] ; last cluster index
0000EC19 752A                   <1>         jne    short mget_w_5
                                <1> 
0000EC1B A3[A4650100]           <1>         mov    [writei.fs_index], eax ; cluster index (for next check)
0000EC20 A1[A8650100]           <1>         mov    eax, [writei.lclust] ; last cluster
0000EC25 EB60                   <1>         jmp    short mget_w_10
                                <1> 
                                <1> mget_w_4: ; 02/11/2016
                                <1>         ; eax = next file position
0000EC27 2B05[88030300]         <1>         sub    eax, [u.count] ; current file position
                                <1>         ; edx = 0
                                <1>         ; ecx = bytes per cluster
0000EC2D F7F1                   <1>         div    ecx
0000EC2F A3[9C650100]           <1>         mov    [writei.c_index], eax ; cluster index
0000EC34 668915[96650100]       <1>         mov    [writei.offset], dx
0000EC3B 66C1EA09               <1>         shr    dx, 9 ; / 512
0000EC3F 8815[8F650100]         <1>         mov    [writei.s_index], dl ; sector index in cluster (0 to spc -1)
                                <1> 
                                <1> mget_w_5:
0000EC45 21C0                   <1>         and    eax, eax ; 0 = First Cluster's index number
0000EC47 750C                   <1>         jnz    short mget_w_6
                                <1> 
0000EC49 A3[A4650100]           <1>         mov    [writei.fs_index], eax ; cluster index (for next check)
0000EC4E A1[A0650100]           <1>         mov    eax, [writei.fclust] ; first cluster
0000EC53 EB32                   <1>         jmp    short mget_w_10
                                <1> 
                                <1> mget_w_6:
```

```
0000EC55 3B05[A4650100]      <1>       cmp    eax, [writei.fs_index] ; current cluster index (>0)
0000EC5B 7507                <1>       jne    short mget_w_7
0000EC5D A1[98650100]        <1>       mov    eax, [writei.cluster] ; current cluster
0000EC62 EB3A                <1>       jmp    short mget_w_11
                             <1> 
                             <1> mget_w_7:
0000EC64 89C1                <1>       mov    ecx, eax
0000EC66 2B0D[A4650100]      <1>       sub    ecx, [writei.fs_index]
0000EC6C 730D                <1>       jnc    short mget_w_8
                             <1>       ; get cluster by index from the first cluster
0000EC6E A1[A0650100]        <1>       mov    eax, [writei.fclust]
0000EC73 8B0D[9C650100]      <1>       mov    ecx, [writei.c_index]
0000EC79 EB05                <1>       jmp    short mget_w_9
                             <1> 
                             <1> mget_w_8:
0000EC7B A1[98650100]        <1>       mov    eax, [writei.cluster] ; beginning cluster
                             <1>       ; ecx = cluster sequence number after the beginning cluster
                             <1>       ; sub  edx, edx ; 0
                             <1> 
                             <1> mget_w_9:
                             <1>       ; EAX = Beginning cluster
                             <1>       ; EDX = Sector index in disk/file section
                             <1>       ;      (Only for SINGLIX file system!)
                             <1>       ; ECX = Cluster sequence number after the beginning cluster
                             <1>       ; ESI = Logical DOS Drive Description Table address
0000EC80 E8D2D8FFFF          <1>       call   get_cluster_by_index
0000EC85 7234                <1>       jc     short mget_w_err ; error code in EAX
                             <1>       ; EAX = Cluster number
                             <1> mget_w_10:
0000EC87 A3[98650100]        <1>       mov    [writei.cluster], eax ; FDT number for Singlix File System
                             <1> 
0000EC8C 807E0300            <1>       cmp    byte [esi+LD_FATType], 0
0000EC90 7638                <1>       jna    short mget_w_13
                             <1>       ; 01/11/2016
0000EC92 8B15[9C650100]      <1>       mov    edx, [writei.c_index]
0000EC98 8915[A4650100]      <1>       mov    [writei.fs_index], edx
                             <1> mget_w_11:
0000EC9E 83E802              <1>       sub    eax, 2
0000ECA1 0FB615[8E650100]    <1>       movzx  edx, byte [writei.spc]
0000ECA8 F7E2                <1>       mul    edx
                             <1> 
0000ECAA 034668              <1>       add    eax, [esi+LD_DATABegin]
0000ECAD 8A15[8F650100]      <1>       mov    dl, [writei.s_index]
0000ECB3 01D0                <1>       add    eax, edx
                             <1> mget_w_12:
0000ECB5 A3[90650100]        <1>       mov    [writei.sector], eax
                             <1>       ;; buffer validation must be done in writei
                             <1>       ;;mov  byte [writei.valid], 1
0000ECBA C3                  <1>       retn
                             <1> 
                             <1> mget_w_err:
0000ECBB A3[C8030300]        <1>       mov    [u.error], eax
0000ECC0 A3[64030300]        <1>       mov    [u.r0], eax
0000ECC5 E9F4D9FFFF          <1>       jmp    error
                             <1> 
                             <1> mget_w_13:
                             <1>       ; EAX = FDT number (Current Section)
                             <1>       ; EDX = Sector index from the first section (0,1,2,3,4...)
0000ECCA 2B15[A4650100]      <1>       sub    edx, [writei.fs_index]
                             <1>       ; EDX = Sector index from current section
0000ECD0 8915[A4650100]      <1>       mov    [writei.fs_index], edx
0000ECD6 40                  <1>       inc    eax ; the first data sector in FS disk section
0000ECD7 01D0                <1>       add    eax, edx
0000ECD9 EBDA                <1>       jmp    short mget_w_12
                             <1> 
                             <1> mget_w_14:
0000ECDB 8A4E12              <1>       mov    cl, [esi+LD_FS_BytesPerSec+1]
0000ECDE D0E9                <1>       shr    cl, 1 ;  ; 1 for 512 bytes, 4 for 2048 bytes
0000ECE0 880D[8E650100]      <1>       mov    [writei.spc], cl  ; sectors per cluster
                             <1>       ; NOTE: writei bytes per sector value is always 512 !
0000ECE6 66C705[94650100]00- <1>       mov    word [writei.bpc], 512
0000ECEE 02                  <1> 
                             <1> 
0000ECEF 89E9                <1>       mov    ecx, ebp
0000ECF1 030D[88030300]      <1>       add    ecx, [u.count] ; next file position
0000ECF7 3B0D[55040300]      <1>       cmp    ecx, [i.size] ; <= file size ?
0000ECFD 0F86C8000000        <1>       jna    mget_w_19 ; no
                             <1> 
0000ED03 29D2                <1>       sub    edx, edx ; 0
0000ED05 8915[90650100]      <1>       mov    [writei.sector], edx ; 0
0000ED0B 668915[96650100]    <1>       mov    [writei.offset], dx  ; byte offset in cluster
0000ED12 8815[8F650100]      <1>       mov    [writei.s_index], dl ; sector index in cluster (0 to spc -1)
                             <1> 
0000ED18 C1E909              <1>       shr    ecx, 9 ; 1 cluster = 512 bytes
0000ED1B 890D[9C650100]      <1>       mov    [writei.c_index], ecx ; section/cluster index
                             <1> 
0000ED21 89D8                <1>       mov    eax, ebx ; FDT number (First FDT address)
                             <1> 
                             <1>       ; is this the 1st mget_w or a next mget_w call ? (by 'writei')
0000ED23 3815[8C650100]      <1>       cmp    byte [writei.valid], dl ; 0
0000ED29 7624                <1>       jna    short mget_w_15
                             <1> 
0000ED2B 8815[8C650100]      <1>       mov    byte [writei.valid], dl ; 0 ; reset ('writei' will set it)
                             <1> 
0000ED31 3B05[A0650100]      <1>       cmp    eax, [writei.fclust]
0000ED37 7516                <1>       jne    short mget_w_15
                             <1> 
0000ED39 8A0D[46030300]      <1>       mov    cl, [cdev]
0000ED3F 3A0D[8D650100]      <1>       cmp    cl, [writei.drv]
0000ED45 7508                <1>       jne    short mget_w_15
                             <1>       ; [writei.l_clust] & [writei.l_index] are valid,
                             <1>       ;  we don't need to get last cluster & last cluster index
0000ED47 8B0D[AC650100]      <1>       mov    ecx, [writei.l_index]
```

```
0000ED4D EB49                <1>       jmp    short mget_w_17
                             <1> mget_w_15:
0000ED4F A3[A0650100]        <1>       mov    [writei.fclust], eax ; first section (FDT number)
                             <1>       ; edx = 0
0000ED54 8915[98650100]      <1>       mov    [writei.cluster], edx ; 0 ; current section
0000ED5A 8915[A4650100]      <1>       mov    [writei.fs_index], edx ; 0 ; curret section index
                             <1>
                             <1>       ; eax = FDT number (section 0 header address)
0000ED60 E81CD8FFFF          <1>       call   get_last_section
0000ED65 0F8250FFFFFF        <1>       jc     mget_w_err ; eax = error code
                             <1>
0000ED6B 8915[A4650100]      <1>       mov    [writei.fs_index], edx ; sector index in last section
                             <1>
0000ED71 A3[A8650100]        <1>       mov    [writei.lclust], eax ; last section address
                             <1>
0000ED76 8B0D[CC630100]      <1>       mov    ecx, [glc_index] ; last section index
0000ED7C 890D[AC650100]      <1>       mov    [writei.l_index], ecx
                             <1>
0000ED82 A0[B0650100]        <1>       mov    al, [writei.ofn]
0000ED87 FEC0                <1>       inc    al
0000ED89 A2[EB650100]        <1>       mov    [setfmod], al ; update lm date&time sign
                             <1>
                             <1> mget_w_16:
                             <1>       ; edx = (existing) last section (sector) index
0000ED8E 8B0D[9C650100]      <1>       mov    ecx, [writei.c_index] ; final section (sector) index
0000ED94 29D1                <1>       sub    ecx, edx
0000ED96 7633                <1>       jna    short mget_w_19
                             <1>        ; ecx = sector count
                             <1> mget_w_17:
0000ED98 A1[A8650100]        <1>       mov    eax, [writei.lclust]
                             <1>       ; ESI = Logical dos drv desc. table address
                             <1>         ; EAX = Last section
                             <1>         ; (ECX = 0 for directory)
                             <1>         ; ECX = sector count (except FDT)
0000ED9D E8A2CDFFFF          <1>       call   add_new_fs_section
0000EDA2 7312                <1>       jnc    short mget_w_18
                             <1>
                             <1>       ; If error number = 27h (insufficient disk space)
                             <1>       ; it is needed to check free consequent sectors
                             <1>       ; (1 data sector at least and +1 section header sector)
                             <1>
0000EDA4 83F827              <1>       cmp    eax, 27h
0000EDA7 0F850EFFFFFF        <1>       jne    mget_w_err ; eax = error code
                             <1>
                             <1>       ; ecx = count of free consequent sectors
                             <1>       ; ecx must be > 1 (1 data + 1 header sector)
0000EDAD 49                  <1>       dec    ecx
0000EDAE 0F8407FFFFFF        <1>       jz     mget_w_err
0000EDB4 EBE2                <1>       jmp    short mget_w_17
                             <1>
                             <1> mget_w_18:
0000EDB6 A3[A8650100]        <1>       mov    [writei.lclust], eax ; (new) last section
                             <1>       ; ecx = sector count (except section header)
0000EDBB 8B15[AC650100]      <1>       mov    edx, [writei.l_index]
0000EDC1 01CA                <1>       add    edx, ecx ; add sector count to index
0000EDC3 8915[AC650100]      <1>       mov    [writei.l_index], edx
0000EDC9 EBC3                <1>       jmp    short mget_w_16
                             <1>
                             <1> mget_w_19:
0000EDCB 89E9                <1>       mov    ecx, ebp
0000EDCD 030D[88030300]      <1>       add    ecx, [u.count]
0000EDD3 890D[55040300]      <1>       mov    [i.size], ecx ; save new file size
                             <1>       ;sub   edx, edx ; 0
                             <1>
0000EDD9 A0[46030300]        <1>       mov    al, [cdev]
0000EDDE A2[8D650100]        <1>       mov    [writei.drv], al ; physical drive number
                             <1>       ; edx = 0
0000EDE3 89E8                <1>       mov    eax, ebp ; file offset
0000EDE5 89C2                <1>       mov    edx, eax
                             <1>       ; 1 cluster = 512 bytes (for Singlix FS)
0000EDE7 C1E809              <1>       shr    eax, 9  ; / 512
0000EDEA 81E2FF010000        <1>       and    edx, 1FFh
                             <1>       ; edx = byte offset in cluster/sector (<= 511)
                             <1>       ; eax = section (sector/cluster) index
0000EDF0 A3[9C650100]        <1>       mov    [writei.c_index], eax
0000EDF5 668915[96650100]    <1>       mov    [writei.offset], dx
                             <1>       ;mov   byte [writei.s_index], 0 ; sector index in cluster
0000EDFC E912FEFFFF          <1>       jmp    mget_w_3
                             <1>
                             <1> update_file_lmdt: ; & update file size
                             <1>       ; 26/10/2016
                             <1>       ; 24/10/2016
                             <1>       ; 23/10/2016
                             <1>       ; 22/10/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ;
                             <1>       ; Update last modification date&time of file
                             <1>       ; (call from syswrite -> writei)
                             <1>       ; ((also updates file size)) // 26/10/2016
                             <1>       ;
                             <1>       ; INPUT:
                             <1>       ;      byte [setfmod] = open file number
                             <1>       ; OUTPUT:
                             <1>       ;      cf = 0 -> success !
                             <1>       ;      cf = 1 -> lmdt update has been failed!
                             <1>       ;
                             <1>       ; Modified registers: eax, ebx, ecx, edx, esi, edi
                             <1>       ;
                             <1>
                             <1>       ;cmp   byte [setfmod], 0
                             <1>       ;jna   short uflmdt_2 ; nothing to do
                             <1>
0000EE01 31C0                <1>       xor    eax, eax
                             <1>
```

```
0000EE03 0FB61D[EB650100]    <1>       movzx  ebx, byte [setfmod]
0000EE0A FECB                <1>       dec    bl ; open file index number (0 based)
                             <1> 
0000EE0C 8AA3[58690100]      <1>       mov    ah, [ebx+OF_DRIVE]
0000EE12 BE00010900          <1>       mov    esi, Logical_DOSDisks
0000EE17 01C6                <1>       add    esi, eax
0000EE19 C0E302              <1>       shl    bl, 2 ; *4
0000EE1C 8B8B[30690100]      <1>       mov    ecx, [ebx+OF_FCLUSTER] ; first cluster
0000EE22 8B93[F8690100]      <1>       mov    edx, [ebx+OF_DIRCLUSTER] ; dir cluster
                             <1> 
0000EE28 D0EB                <1>       shr    bl, 1 ; /2
0000EE2A 0FB7BB[986A0100]    <1>       movzx  edi, word [ebx+OF_DIRENTRY]
                             <1> 
0000EE31 803D[28610100]01    <1>       cmp    byte [DirBuff_ValidData], 1
0000EE38 726E                <1>       jb     short uflmdt_4
                             <1> 
0000EE3A A0[26610100]        <1>       mov    al, [DirBuff_DRV]
0000EE3F 2C41                <1>       sub    al, 'A'
0000EE41 38E0                <1>       cmp    al, ah
0000EE43 7563                <1>       jne    short uflmdt_4 ; different drive
0000EE45 8A4603              <1>       mov    al, [esi+LD_FATType]
0000EE48 3A05[27610100]      <1>       cmp    al, [DirBuff_FATType]
0000EE4E 755B                <1>       jne    short uflmdt_5 ; different FS type
0000EE50 3B15[2D610100]      <1>       cmp    edx, [DirBuff_Cluster]
0000EE56 7553                <1>       jne    short uflmdt_5 ; different cluster
                             <1> 
                             <1> uflmdt_1:
                             <1>       ; Directory buffer is ready here!
                             <1>       ; OF_FCLUSTER must be compared/verified
0000EE58 BE00000800          <1>       mov    esi, Directory_Buffer
0000EE5D 66C1E705            <1>       shl    di, 5 ; dir entry index * 32
0000EE61 01FE                <1>       add    esi, edi ; offset
                             <1>       ;
0000EE63 F6460B18            <1>       test   byte [esi+DirEntry_Attr], 18h ; Vol & Dir
0000EE67 750F                <1>       jnz    short uflmdt_2 ; not a valid file !
0000EE69 668B4614            <1>       mov    ax, [esi+DirEntry_FstClusHI]
0000EE6D C1E010              <1>       shl    eax, 16
0000EE70 668B461A            <1>       mov    ax, [esi+DirEntry_FstClusLO]
0000EE74 39C8                <1>       cmp    eax, ecx ; same first cluster ?
0000EE76 7407                <1>       je     short uflmdt_3 ; yes, it is OK !!!
                             <1> 
                             <1> uflmdt_2:
                             <1>       ; save directory buffer if has modified/changed sign
                             <1>       ; (It is good to save dir buff even if the searched
                             <1>       ; directory entry is not found !?)
0000EE78 E81BBAFFFF          <1>       call   save_directory_buffer
0000EE7D F9                  <1>       stc    ; update failed
0000EE7E C3                  <1>       retn
                             <1> 
                             <1> uflmdt_3:
                             <1>       ; Update directory entry
                             <1>       ; 26/10/2016
0000EE7F D0E3                <1>       shl    bl, 1 ; *2
0000EE81 8B83[A8690100]      <1>       mov    eax, [ebx+OF_SIZE] ; file size
0000EE87 89461C              <1>       mov    [esi+DirEntry_FileSize], eax
                             <1>       ;
0000EE8A E86BB9FFFF          <1>       call   convert_current_date_time
                             <1>       ; OUTPUT -> DX = Date in dos dir entry format
                             <1>         ;       AX = Time in dos dir entry format
0000EE8F 66894616            <1>       mov    [esi+DirEntry_WrtTime], ax
0000EE93 66895618            <1>       mov    [esi+DirEntry_WrtDate], dx
0000EE97 66895612            <1>       mov    [esi+DirEntry_LastAccDate], dx
0000EE9B C605[28610100]02    <1>       mov    byte [DirBuff_ValidData], 2
0000EEA2 E8F1B9FFFF          <1>       call   save_directory_buffer
0000EEA7 C3                  <1>       retn
                             <1> 
                             <1> uflmdt_4:
                             <1>       ; Directory buffer sector read&write
                             <1>       ; 23/10/2016
                             <1>       ;
0000EEA8 8A4603              <1>       mov    al, [esi+LD_FATType]
                             <1> uflmdt_5:
0000EEAB BB[9C090300]        <1>       mov    ebx, rw_buffer ; Common r/w sector buffer addr
                             <1> 
0000EEB0 20C0                <1>       and    al, al ; 0 = Singlix FS
0000EEB2 0F8492000000        <1>       jz     uflmdt_11
                             <1> 
0000EEB8 21D2                <1>       and    edx, edx
0000EEBA 7521                <1>       jnz    short uflmdt_9
                             <1> 
0000EEBC 3C02                <1>       cmp    al, 2  ; 3 = FAT32
0000EEBE 771A                <1>       ja     short uflmdt_8
                             <1> 
0000EEC0 89F8                <1>       mov    eax, edi ; directory entry index number
0000EEC2 66C1E804            <1>       shr    ax, 4 ; 16 entries per sector
0000EEC6 034664              <1>       add    eax, [esi+LD_ROOTBegin]
                             <1>       ; eax = root directory sector
                             <1> uflmdt_6:
0000EEC9 50                  <1>       push   eax ; * ; disk sector address
0000EECA 51                  <1>       push   ecx ; first cluster
0000EECB B901000000          <1>       mov    ecx, 1
                             <1>       ; ecx = sector count
0000EED0 E803090000          <1>       call   disk_read
0000EED5 59                  <1>       pop    ecx
0000EED6 731A                <1>       jnc    short uflmdt_10
0000EED8 58                  <1>       pop    eax ; *
                             <1> uflmdt_7:
0000EED9 C3                  <1>       retn
                             <1> 
                             <1> uflmdt_8:
0000EEDA 8B5632              <1>       mov    edx, [esi+LD_BPB+FAT32_RootFClust]
                             <1> uflmdt_9:
0000EEDD 83FA02              <1>       cmp    edx, 2
```

```
0000EEE0 72F7                <1>       jb    short uflmdt_7 ; invalid, nothing to do
                             <1>
0000EEE2 83EA02              <1>       sub   edx, 2
0000EEE5 89D0                <1>       mov   eax, edx
0000EEE7 0FB65613            <1>       movzx edx, byte [esi+LD_BPB+SecPerClust]
0000EEEB F7E2                <1>       mul   edx
0000EEED 034668              <1>       add   eax, [esi+LD_DATABegin]
                             <1>       ; eax = sub directory (data) sector
0000EEF0 EBD7                <1>       jmp   short uflmdt_6
                             <1>
                             <1> uflmdt_10:
                             <1>       ; Directory sector buffer is ready here!
                             <1>       ; OF_FCLUSTER must be compared/verified
                             <1>       ; edi = dir entry index number (<= 2047)
0000EEF2 6683E70F            <1>       and   di, 0Fh ; 16 entries per sector
0000EEF6 66C1E705            <1>       shl   di, 5 ; dir entry index * 32
0000EEFA 81C7[9C090300]      <1>       add   edi, rw_buffer
                             <1>       ;
0000EF00 F6470B18            <1>       test  byte [edi+DirEntry_Attr], 18h ; Vol & Dir
0000EF04 0F856EFFFFFF        <1>       jnz   uflmdt_2 ; not a valid file !
0000EF0A 668B5714            <1>       mov   dx, [edi+DirEntry_FstClusHI]
0000EF0E C1E210              <1>       shl   edx, 16
0000EF11 668B571A            <1>       mov   dx, [edi+DirEntry_FstClusLO]
0000EF15 39CA                <1>       cmp   edx, ecx ; same first cluster ?
0000EF17 0F855BFFFFFF        <1>       jne   uflmdt_2 ; no !?
                             <1>
                             <1>       ; Update directory entry
0000EF1D E8D8B8FFFF          <1>       call  convert_current_date_time
                             <1>       ; OUTPUT -> DX = Date in dos dir entry format
                             <1>         ;        AX = Time in dos dir entry format
0000EF22 66894716            <1>       mov   [edi+DirEntry_WrtTime], ax
0000EF26 66895718            <1>       mov   [edi+DirEntry_WrtDate], dx
0000EF2A 66895712            <1>       mov   [edi+DirEntry_LastAccDate], dx
                             <1>
0000EF2E 58                  <1>       pop   eax ; *
                             <1>
0000EF2F BB[9C090300]        <1>       mov   ebx, rw_buffer ; Common r/w sector buffer addr
0000EF34 B901000000          <1>       mov   ecx, 1
                             <1>       ; esi = logical dos description table address
                             <1>       ; eax = disk sector number/address (LBA)
                             <1>       ; ecx = sector count
                             <1>       ; ebx = buffer address
0000EF39 E88B080000          <1>       call  disk_write
0000EF3E 0F8234FFFFFF        <1>       jc    uflmdt_2
                             <1>
                             <1>       ; save directory buffer if has modified/changed sign
0000EF44 E84FB9FFFF          <1>       call  save_directory_buffer
0000EF49 C3                  <1>       retn
                             <1>
                             <1> uflmdt_11:
                             <1>       ; 24/10/2016
                             <1>       ; Update last modification date & time of a file
                             <1>       ; on a disk with Singlix File System.
                             <1>       ;
                             <1>       ; (Method: Read the FDT -File Description Table-
                             <1>       ; sector of the file and update the lmdt data fields,
                             <1>       ; then write FDT sector to the disk.
                             <1>       ; /// It is easy but there is compatibility buffer
                             <1>       ; method also for changing directory entry data and
                             <1>       ; also there are some programming issues for Singlix
                             <1>       ; file system (TRFS), which are not completed yet!)
                             <1>       ;
                             <1>       ; Not ready yet ! (24/10/2016)
                             <1>       ; /// Temporary code for error return ! ///
0000EF4A 31C0                <1>       xor   eax, eax
0000EF4C F9                  <1>       stc
0000EF4D C3                  <1>       retn
                             <1>
                             <1> sysalloc:
                             <1>       ; 14/10/2017
                             <1>       ; 20/08/2017, 01/09/2017
                             <1>       ; 20/02/2017, 04/03/2017, 15/05/2017
                             <1>       ; 19/02/2017 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; (TRDOS 386 feature only!)
                             <1>       ;
                             <1>       ; Allocate Contiguous Memory Block/Pages (for user)
                             <1>       ; (System call for DMA Buffer allocation etc.)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      EBX = Virtual address (for user)
                             <1>       ;           (Physical memory block/aperture
                             <1>       ;           will be mapped to this virtual address)
                             <1>       ;      ECX = Byte Count
                             <1>       ;           (will be rounded up to page border)
                             <1>       ;      If ECX = 0
                             <1>       ;          System call will return with an error (cf=1)
                             <1>       ;          but ECX will contain maximum size of
                             <1>       ;          available memory aperture and physical
                             <1>       ;          (beginning) address of that aperture
                             <1>       ;          (which have maximum size) will be in EAX.
                             <1>       ;      EDX = Upper limit of the requested physical memory
                             <1>       ;            block/pages.
                             <1>       ;           (The last byte address of the memory aperture
                             <1>       ;            must not be equal to or above this limit.)
                             <1>       ;      If EDX = 0
                             <1>       ;         there is NOLIMIT !
                             <1>       ;      If EDX = 0FFFFFFFFh (-1)
                             <1>       ;         ESI = Lower Limit !
                             <1>       ;            (Beginning of the block must not be 'less'
                             <1>       ;            than this.) (Must be equal to or above...)
                             <1>       ;         EDI = Upper Limit !
                             <1>       ;            (End of the block must be !less! than this)
                             <1>       ;            (The last byte addr of the memory aperture
                             <1>         ;             must not be equal to or above this limit.)
                             <1>         ;
                             <1>         ; OUTPUT ->
                             <1>         ;      If CF = 0
                             <1>         ;      EAX = Physical address of the allocated memory block
                             <1>         ;      ECX = Allocated bytes (as rounded up to page borders)
                             <1>         ;      EBX = Virtual address (as rounded up)
                             <1>         ;      IF CF = 1
                             <1>         ;          Requested (size of) Memory block could not be
                             <1>         ;          allocated to the user!
                             <1>         ;      IF CF = 1 & EAX = 0 (Insufficient memory error!)
                             <1>         ;         ECX = Total number of free bytes
                             <1>         ;               (not size of available contiguous bytes!)
                             <1>         ;      If CF = 1 & EAX > 0
                             <1>         ;         there is not a memory aperture with requested size
                             <1>         ;         but total free mem is not less than requested size.
                             <1>         ;         EAX = Physical addr of available memory aperture
                             <1>         ;              with max size
                             <1>         ;               (but it doesn't fit to the conditions!)
                             <1>         ;         ECX = Size of available memory aperture in bytes.
                             <1>         ;      If CF = 1 -> EAX = 0FFFFFFFFh
                             <1>         ;         Conditions/Parameters are wrong !
                             <1>         ;         ECX is same with input value.
                             <1>         ;
                             <1>         ; Note:      Previously allocated pages will be deallocated if
                             <1>         ;       new allocation conditions are met.
                             <1>         ;
                             <1>         ; Note: u.break control may be included in future versions
                             <1>         ;
                             <1>
0000EF4E 31C0                     <1>         xor    eax, eax ; 0
                             <1>         ; 14/10/2017
0000EF50 4A                       <1>         dec    edx ; is there a limit ?
0000EF51 7810                     <1>         js     short sysalloc_1 ;  0 -> 0FFFFFFFFh -> NO LIMIT
0000EF53 42                       <1>         inc    edx ; > 0
                             <1>         ; Check upper address limit
                             <1>         ;(round up to page borders)
0000EF54 81C1FF0F0000             <1>         add    ecx, PAGE_SIZE-1 ; 4095
0000EF5A 6681E100F0               <1>         and    cx, ~PAGE_OFF ; not 4095
0000EF5F 39CA                     <1>         cmp    edx, ecx ; upper limit - block size
0000EF61 7224                     <1>         jb     short sysalloc_err
                             <1> sysalloc_1:
                             <1>         ; EAX = Beginning address (physical)
                             <1>         ; EAX = 0 -> Allocate mem block from the 1st proper aperture
                             <1>         ; ECX = Number of bytes to be allocated
0000EF63 E8BC64FFFF               <1>         call   allocate_memory_block
0000EF68 721D                     <1>         jc     short sysalloc_err
                             <1>         ; 01/09/2017
0000EF6A 29C2                     <1>         sub    edx, eax ; upper limit address - beginning address
0000EF6C 760F                     <1>         jna    short sysalloc_3 ; begin addr not less than the limit
0000EF6E 39CA                     <1>         cmp    edx, ecx
0000EF70 720B                     <1>         jb     short sysalloc_3 ; end address overs the limit
                             <1> sysalloc_2:
                             <1>         ; EAX = Beginning (physical) addr of the allocated mem block
                             <1>         ; ECX = Num of allocated bytes (rounded up to page borders)
0000EF72 50                       <1>         push   eax ; * ; 04/03/2017
                             <1>         ; Here, requested contiquous memory pages have been allocated
                             <1>         ; on Memory Allocation Table but user's page directory
                             <1>         ; and page tables have not been updated yet!
0000EF73 51                       <1>         push   ecx ; **
                             <1>         ; ebx = virtual address (will be rounded up to page border)
                             <1>         ; ecx = number of bytes to be deallocated
                             <1>         ;      will be adjusted to ebx+ecx round down - ebx round up
0000EF74 E80668FFFF               <1>         call   deallocate_user_pages
0000EF79 731F                     <1>         jnc    short sysalloc_4 ; EAX = Deallocated memory bytes
0000EF7B 59                       <1>         pop    ecx ; **
0000EF7C 58                       <1>         pop    eax ; *
                             <1> sysalloc_3:
                             <1>         ; error !
                             <1>         ; restore Memory Allocation Table Content
0000EF7D E8AF66FFFF               <1>         call   deallocate_memory_block
0000EF82 31C0                     <1>         xor    eax, eax ; 0
0000EF84 48                       <1>         dec    eax ; 0FFFFFFFFh ; 15/05/2017
0000EF85 EB09                     <1>         jmp    short sysalloc_wrong
                             <1> sysalloc_err:
0000EF87 8B2D[60030300]           <1>         mov    ebp, [u.usp]  ; ebp points to user's registers
0000EF8D 894D18                   <1>         mov    [ebp+24], ecx ; return to user with ecx value
                             <1> sysalloc_wrong:
                             <1>         ; eax = 0FFFFFFFFh
0000EF90 A3[64030300]             <1>         mov    [u.r0], eax
0000EF95 E924D7FFFF               <1>         jmp    error
                             <1> sysalloc_4:
0000EF9A 8B2D[60030300]           <1>         mov    ebp, [u.usp]  ; ebp points to user's registers
0000EFA0 894518                   <1>         mov    [ebp+24], eax ; return to user with ecx value
0000EFA3 895D10                   <1>         mov    [ebp+16], ebx ; new value of ebx (rounded up)
0000EFA6 89C1                     <1>         mov    ecx, eax ; byte count (from 'deallocate_user_pages')
0000EFA8 5A                       <1>         pop    edx ; ** ; discard (another) byte count
0000EFA9 58                       <1>         pop    eax ; *
0000EFAA A3[64030300]             <1>         mov    [u.r0], eax ; physical address
                             <1>
0000EFAF 51                       <1>         push   ecx ; 20/08/2017
                             <1>         ;
                             <1>         ; Write newly allocated contiguous (physical) pages
                             <1>         ; on page dir and page tables of current user/process
                             <1>         ; as PRESENT, USER, WRITABLE
                             <1>         ; (then clear allocated pages)
0000EFB0 E8BF68FFFF               <1>         call   allocate_user_pages
                             <1>         ;jnc   sysret ; OK! return to process with success...
                             <1>
                             <1>         ; 20/08/2017 ('sysdma' modification)
0000EFB5 59                       <1>         pop    ecx
0000EFB6 A1[64030300]             <1>         mov    eax, [u.r0]   ; physical address (of the block)
                             <1>
```

```
0000EFBB 721D                <1>       jc     short sysalloc_6
                             <1>
0000EFBD 833D[00700100]FF    <1>       cmp    dword [dma_addr], 0FFFFFFFFh ; -1
0000EFC4 0F8214D7FFFF        <1>       jb     sysret
                             <1>
0000EFCA A3[00700100]        <1>       mov    [dma_addr], eax ; save dma address for sysdma
0000EFCF 890D[04700100]      <1>       mov    [dma_size], ecx ; save dma buff size for sysdma
                             <1>
0000EFD5 E904D7FFFF          <1>       jmp    sysret
                             <1>
                             <1> sysalloc_6:
                             <1>       ;
                             <1>       ; unexpected error ! insufficient memory !? conflict !?
                             <1>       ; (!!?there is not a free page for a new page table?!!)
                             <1>       ; We need to terminate process with error message !!!
                             <1>       ;
0000EFDA 8B2D[60030300]      <1>       mov    ebp, [u.usp]  ; ebp points to user's registers
0000EFE0 8B4D18              <1>       mov    ecx, [ebp+24] ; byte count
                             <1>
                             <1>       ; 20/08/2017
                             <1>       ;mov   eax, [u.r0]   ; physical address (of the block)
                             <1>
                             <1>       ;
                             <1>       ; restore Memory Allocation Table Content
0000EFE3 E84966FFFF          <1>       call   deallocate_memory_block
                             <1>       ;
0000EFE8 803D[C25E0000]03    <1>       cmp    byte [CRT_MODE], 3 ; 80x25 text mode?
0000EFEF 7407                <1>       je     short sysalloc_7 ; yes
                             <1>       ; Current mode is VGA (or CGA graphics) mode,
                             <1>       ; We need to return to text mode for displaying
                             <1>       ; error message just before 'sysexit'.
0000EFF1 B003                <1>       mov    al, 3
0000EFF3 E86D25FFFF          <1>       call   _set_mode
                             <1> sysalloc_7:
0000EFF8 BE[34100100]        <1>       mov    esi, beep_Insufficient_Memory ; error message
0000EFFD E85B73FFFF          <1>       call   print_msg ; print/display the message
0000F002 B801000000          <1>       mov    eax, 1 ; ax=1 is needed for 'sysexit' procedure
0000F007 E959D8FFFF          <1>       jmp    sysexit ; and terminate the process !
                             <1>
                             <1> sysdalloc:
                             <1>       ; 19/02/2017 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; (TRDOS 386 feature only!)
                             <1>       ;
                             <1>       ; Deallocate Memory Block/Pages (for user)
                             <1>       ; (Complementary call for sysalloc.)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      EBX = Virtual address (for user)
                             <1>       ;            (will be rounded up to page border)
                             <1>       ;      ECX = Byte Count
                             <1>       ;           (will be adjusted to page borders)
                             <1>       ;      If ICX = 0
                             <1>       ;         nothing to do
                             <1>       ;      If EBX + ECX > User's ESP
                             <1>       ;         nothing to do
                             <1>       ;
                             <1>       ; Note: u.break control may be included in future versions
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      If CF = 0
                             <1>       ;         EAX = Deallocated memory bytes
                             <1>       ;         EBX = Virtual address (as rounded up)
                             <1>       ;      IF CF = 1
                             <1>       ;         EAX = 0
                             <1>       ;
                             <1>       ; Note:      Main purpose of this call is to deallocate/release
                             <1>       ;      previously allocated (physically) contiguous memory
                             <1>       ;      pages but beginning (virtual) address may not be
                             <1>       ;      followed by physically contiguous pages. So, this
                             <1>       ;      system call will deallocate user's virtually
                             <1>       ;      contiguous memory pages. Also, there is not any
                             <1>       ;      objections to use this system call without sysalloc
                             <1>       ;      system call; only possible objection is to lost data
                             <1>       ;      within user's memory space, if the beginning address
                             <1>       ;      and size is not proper.
                             <1>       ;
                             <1>       ; Note: Empty page tables will not be deallocated!!!
                             <1>       ;       (they will be deallocated at process termination)
                             <1>       ;
                             <1>       ; Note: When the program terminates itself or when it is
                             <1>       ;      terminated by operating system kernel, all allocated
                             <1>       ;      memory pages will be deallocated during termination
                             <1>       ;      stage. So, 'sysdalloc' is not necessary except
                             <1>       ;      forgiving memory block to other programs/processes.
                             <1>       ;
0000F00C 8B15[5C030300]      <1>       mov    edx, [u.sp]
0000F012 8B420C              <1>       mov    eax, [edx+12] ; user's stack pointer
0000F015 29C8                <1>       sub    eax, ecx ; esp - byte count
0000F017 24FC                <1>       and    al, 0FCh ; dword alignment
0000F019 39D8                <1>       cmp    eax, ebx
0000F01B 7220                <1>       jb     short sysdalloc_err ; deallocation overlaps with stack
                             <1>
0000F01D 31C0                <1>       xor    eax, eax
0000F01F 21C9                <1>       and    ecx, ecx
0000F021 7407                <1>       jz     short sysdalloc_2
                             <1>
0000F023 E85767FFFF          <1>       call   deallocate_user_pages
0000F028 7213                <1>       jc     short sysdalloc_err
                             <1>
                             <1> sysdalloc_2:
0000F02A A3[64030300]        <1>       mov    [u.r0], eax
0000F02F 8B2D[60030300]      <1>       mov    ebp, [u.usp]
0000F035 895D10              <1>       mov    [ebp+16], ebx ; new value of ebx
```

```
0000F038 E9A1D6FFFF               <1>       jmp    sysret
                                  <1> 
                                  <1> sysdalloc_err:
0000F03D A3[64030300]             <1>       mov    [u.r0], eax ; 0
0000F042 E977D6FFFF               <1>       jmp    error
                                  <1> 
                                  <1> syscalbac:
                                  <1>       ; SYS CALLBACK
                                  <1>       ; 16/04/2017
                                  <1>       ; 14/04/2017
                                  <1>       ; 13/04/2017
                                  <1>       ; 28/02/2017
                                  <1>       ; 26/02/2017
                                  <1>       ; 24/02/2017
                                  <1>       ; 21/02/2017 - TRDOS 386 (TRDOS v2.0)
                                  <1>       ; (TRDOS 386 feature only!)
                                  <1>       ;
                                  <1>       ; Link or unlink IRQ callback service to/from user (ring 3)
                                  <1>       ;
                                  <1>       ; INPUT ->
                                  <1>       ;     BL = IRQ number (Hardware interrupt request number)
                                  <1>       ;          (0 t0 15 but IRQ 0,1,2,6,8,14,15 are prohibited)
                                  <1>       ;          IRQ numbers 3,4,5,7,9,10,11,12,13 are valid
                                  <1>       ;          (numbers >15 are invalid)
                                  <1>       ;
                                  <1>       ;     BH = 0 = Unlink IRQ (in BL) from user (ring 3) service
                                  <1>       ;          1 = Link IRQ by using Signal Response Byte method
                                  <1>       ;          2 = Link IRQ by using Callback service method
                                  <1>       ;          3 = Link IRQ by using Auto Increment S.R.B. method
                                  <1>       ;         >3 = invalid
                                  <1>       ;
                                  <1>       ;     CL = Signal Return/Response Byte value
                                  <1>       ;
                                  <1>       ;     If BH = 2, kernel will put a counter value
                                  <1>       ;               (into the S.R.B. addr)
                                  <1>       ;              between 0 to 255. (start value = CL+1)
                                  <1>       ;
                                  <1>       ;     NOTE: counter value, for example: even and odd numbers
                                  <1>       ;           may be used for -audio- DMA buffer switch
                                  <1>       ;           within double buffer method, etc.
                                  <1>       ;
                                  <1>       ;     EDX = Signal return (Response) byte address
                                  <1>       ;                           - or -
                                  <1>       ;           Interrupt/Callback service/routine address
                                  <1>       ;
                                  <1>       ;           (virtual address in user's memory space)
                                  <1>       ;
                                  <1>       ; OUTPUT ->
                                  <1>       ;     CF = 0 & EAX = 0 -> Successful setting
                                  <1>       ;     CF = 1 & EAX > 0 -> IRQ is prohibited or locked
                                  <1>       ;                   by another process
                                  <1>       ;             eax = ERR_PERM_DENIED -> prohibited or locked
                                  <1>       ;             eax = ERR_INV_PARAMETER ->
                                  <1>       ;                   invalid parameter/option or bad address
                                  <1>       ;
                                  <1>       ;     NOTE: Timer callbacks are set by using 'systimer'
                                  <1>       ;           system call (IRQ 0, PIT and IRQ 8, RTC)
                                  <1>       ;
                                  <1>       ;           Direct keyboard access is performed by using
                                  <1>       ;           Keyboard Interrupt (INT 32h)
                                  <1>       ;
                                  <1>       ;           It is prohibited here because:
                                  <1>       ;            1) Signal Response Byte method has not advantage
                                  <1>       ;               against INT 32h, function AH = 1. Also,
                                  <1>       ;               keyboard service interrupt will return with
                                  <1>       ;               ascii and scan codes (AL, AH) while
                                  <1>       ;               SRB method has only 1 byte space for ascii code
                                  <1>       ;               or scan code. One byte signal response is used
                                  <1>       ;               for ensuring very simple and very fast
                                  <1>       ;               virtual to physical memory address conversion
                                  <1>       ;               without any memory page crossover risk.
                                  <1>       ;               (Otherwise double page conversion or word
                                  <1>       ;               alignment would be needed.)
                                  <1>       ;            2) Badly written user code (callback code)
                                  <1>       ;               can prevent keyboard and timesharing functions
                                  <1>       ;               of the operating system via continuous and long
                                  <1>       ;               keyboard event handling by callback service.
                                  <1>       ;               (It can cause to lose immediate keystroke
                                  <1>       ;               response from hardware to user.)
                                  <1>       ;            3) If user will check any keyboard events, 'getkey'
                                  <1>       ;               (or 'getchar') must have more priority than other
                                  <1>       ;               (video etc.) events because only control ability
                                  <1>       ;               on a procedural infinite loop is a keyboard or
                                  <1>       ;               mouse event. So user can use keyboard function
                                  <1>       ;               at the end or at the beginning of a loop.
                                  <1>       ;               In this case, INT 32h is used for that purpose
                                  <1>       ;               and timer interrupt etc. callbacks can be used
                                  <1>       ;               for dynamic and synchronized data refresh/transfer
                                  <1>       ;               while cpu is in a static loop (without polling).
                                  <1>       ;               Keyboard Int callback is not more useful because
                                  <1>       ;               already a manual check (a key is pressed or not)
                                  <1>       ;               can be performed (via INT 32h, AH = 1) efficiently
                                  <1>       ;               in a loop to prevent a locked infinitive loop.
                                  <1>       ;
                                  <1>       ;         Disk IRQs (6,14,15) have been phohibited from ring 3
                                  <1>       ;         callback because, disk operations (file system services
                                  <1>       ;         etc.) are independent from user program, for fast disk r/w.
                                  <1>       ;         They are not more useful at ring 3 while they are in use
                                  <1>       ;         by standard diskio functions which are mandatory part of
                                  <1>       ;         (monolithic) OS kernel and mainprog command interpreter.
                                  <1>       ;         INT 33h diskio functions are enough for user level disk
                                  <1>       ;         r/w.
                                  <1>       ;
```

```
                             <1>       ; TRDOS 386 - IRQ CALLBACK structures (parameters):
                             <1>       ;
                             <1>       ;         [u.irqlock] = 1 word, IRQ flags (0-15) that indicates
                             <1>       ;                   which IRQs are locked by (that) user.
                             <1>       ;                    Lock and unlock (by user) will change
                             <1>       ;                   these flags or 'terminate process' (sysexit)
                             <1>       ;                   will clear these flags and unlock those IRQs.
                             <1>       ;
                             <1>       ;                   Bit 0 is for IRQ 0 and Bit 15 is for IRQ 15
                             <1>       ;
                             <1>       ;         IRQ(x).owner      : 1 byte, user, [u.uno], 0 = free (unlocked)
                             <1>       ;
                             <1>       ;         IRQ(x).method : 1 byte for callback method & status
                             <1>       ;                      0 = Signal Response Byte method
                             <1>       ;                      1 = Callback service method
                             <1>       ;                      >1 = invalid for current 'syscalback'.
                             <1>       ;                   or(+) 80h = IRQ is in use by system (ring 0)
                             <1>       ;                               function (audio etc.) or
                             <1>       ;                             a device driver.
                             <1>       ;                   (system function will ignore the lock/owner)
                             <1>       ;
                             <1>       ;         IRQ(x).srb: 1 byte, Signal Return/Response byte value
                             <1>       ;                     (a fixed value by user or a counter value
                             <1>       ;                    from 0 to 255, which is increased by every
                             <1>       ;                    interrupt just before putting it into
                             <1>       ;                    the Signal Response byte address
                             <1>       ;                    (This is not used in callback serv method)
                             <1>       ;
                             <1>       ;         IRQ(x).addr      : 1 dword
                             <1>       ;                     Signal Response Byte address (physical)
                             <1>       ;                                -or-
                             <1>       ;                     Callback service address (virtual)
                             <1>       ;
                             <1>       ;         IRQ(x).dev: 1 byte
                             <1>       ;                     0 = Default device or kernel function
                             <1>       ;                                -or-
                             <1>       ;                     1-255 = Assigned device driver number
                             <1>       ;
                             <1>       ;         (x) = 3,4,5,7,9,10,11,12,13
                             <1>       ;
                             <1>       ;
                             <1>       ;      NOTE: If user's process/program calls the kernel (INT 40h)
                             <1>       ;            while it is already running in a (ring 3) callback
                             <1>       ;            service, kernel will force (convert) system call to
                             <1>       ;            'sysrele' (sys release). So, this feature provides
                             <1>       ;            easy and simple usage of callback services without
                             <1>       ;            falling into deepless <please 'callback me' then
                             <1>       ;            let me 'callback you'> cycles! (User must return
                             <1>       ;            from callback service by using 'sysrele' system
                             <1>       ;            call, without a significant delay. Otherwise user
                             <1>       ;            process/program may be late to catch the next event
                             <1>       ;            within same callback purpose.
                             <1>       ;
                             <1>
0000F047 30C0                <1>       xor    al, al ; the caller is 'syscalbac' sign/flag
0000F049 E85A180000          <1>       call   set_irq_callback_service
                             <1>       ; 16/04/2017
0000F04E A3[64030300]        <1>       mov    [u.r0], eax
0000F053 0F8385D6FFFF        <1>       jnc    sysret
0000F059 A3[C8030300]        <1>       mov    dword [u.error], eax
0000F05E E95BD6FFFF          <1>       jmp    error
                             <1>
                             <1> sysfpstat:
                             <1>       ; 28/02/2017 - TRDOS 386 (TRDOS v2.0)
                             <1>       ; (TRDOS 386 feature only!)
                             <1>       ;
                             <1>       ; Set or reset FPU registers save/restore option (for user)
                             <1>       ;             (during software task switching, wswap-rswap)
                             <1>       ;
                             <1>       ; INPUT ->
                             <1>       ;      BL = 0 -> reset
                             <1>       ;      BL = 1 -> set (FPU register will be saved and restored)
                             <1>       ;
                             <1>       ; OUTPUT ->
                             <1>       ;      cf = 0 -> no error, FPU is ready...
                             <1>       ;               (EAX = 0)
                             <1>       ;      Cf = 1 -> error, 80387 FPU is not ready !
                             <1>       ;               (EAX = 0FFFFFFFFh)
                             <1>
0000F063 31C0                <1>       xor    eax, eax
0000F065 803D[F8650100]00    <1>       cmp    byte [fpready], 0
0000F06C 7613                <1>       jna    short sysfpstat_err
                             <1>
0000F06E 80E301              <1>       and    bl, 1 ; use BIT 0 only !
0000F071 881D[DA030300]      <1>       mov    [u.fpsave], bl
0000F077 A3[64030300]        <1>       mov    [u.r0], eax ; 0
0000F07C E95DD6FFFF          <1>       jmp    sysret
                             <1>
                             <1> sysfpstat_err:
0000F081 48                  <1>       dec    eax ; 0FFFFFFFFh
0000F082 A3[64030300]        <1>       mov    [u.r0], eax ; -1
0000F087 E932D6FFFF          <1>       jmp    error
                             <1>
                             <1> sysdelete: ; Delete (Remove, Unlink) File
                             <1>       ; 29/12/2017 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>         ; INPUT ->
                             <1>         ;          EBX = File name (ASCIIZ string) address
                             <1>       ; OUTPUT ->
                             <1>       ;          cf = 0 -> eax = 0
                             <1>       ;          cf = 1 -> Error code in AL
                             <1>       ;
                             <1>       ; Modified Registers: EAX (at the return of system call)
                             <1>       ;
                             <1>
0000F08C 89DE                <1>       mov    esi, ebx
                             <1>       ; file name is forced, change directory as temporary
                             <1>       ;mov   ax, 1
                             <1>       ;mov   [FFF_Valid], ah ; 0 ; reset
                             <1>       ;call  set_working_path
0000F08E E8680B0000          <1>       call   set_working_path_x
0000F093 731D                <1>       jnc    short sysdelete_1
                             <1>
0000F095 21C0                <1>       and    eax, eax  ; 0 -> Bad Path!
0000F097 7505                <1>       jnz    short sysdelete_err
                             <1>       ; eax = 0
                             <1> sysdelete_path_err:
0000F099 B813000000          <1>       mov    eax, ERR_INV_PATH_NAME ; 'bad path name !'
                             <1> sysdelete_err:
0000F09E A3[64030300]        <1>       mov    [u.r0], eax
0000F0A3 A3[C8030300]        <1>       mov    [u.error], eax
0000F0A8 E8230C0000          <1>       call   reset_working_path
0000F0AD E90CD6FFFF          <1>       jmp    error
                             <1> sysdelete_1:
                             <1>       ;mov   esi, FindFile_Name
0000F0B2 66B80018            <1>       mov    ax, 1800h ; Only files
0000F0B6 E8A891FFFF          <1>       call   find_first_file
0000F0BB 72E1                <1>       jc     short sysdelete_err
                             <1> sysdelete_2:
                             <1>       ; check file attributes
                             <1>
                             <1>       ;test  bl, 17 ; system, hidden, readonly, directory
0000F0BD F6C307              <1>         test bl, 7 ; system, hidden, readonly
0000F0C0 7407                <1>         jz   short sysdelete_3
                             <1>
0000F0C2 B80B000000          <1>         mov  eax, ERR_FILE_ACCESS ; 11 = 'permission denied !'
0000F0C7 EBD5                <1>         jmp  short sysdelete_err
                             <1> sysdelete_3:
0000F0C9 6621D2              <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
0000F0CC 7407                <1>       jz     short sysdelete_4
0000F0CE B81A000000          <1>       mov    eax, ERR_INV_FILE_NAME ; 26 = 'invalid file name !'
0000F0D3 EBC9                <1>         jmp  short sysdelete_err
                             <1> sysdelete_4:
                             <1>       ;mov   bh, [LongName_EntryLength]
0000F0D5 883D[6A630100]      <1>       mov    [DelFile_LNEL], bh ; Long name entry length (if > 0)
                             <1>       ; edi = Directory Entry Offset (DirBuff)
                             <1>       ; esi = Directory Entry (FFF Structure)
0000F0DB E800BBFFFF          <1>       call   remove_file
0000F0E0 72BC                <1>       jc     short sysdelete_err
                             <1> sysrmdir_5:
0000F0E2 31C0                <1>       xor    eax, eax ; 0
0000F0E4 A3[64030300]        <1>       mov    [u.r0], eax
                             <1>       ;mov   [u.error], eax
0000F0E9 E8E20B0000          <1>       call   reset_working_path
0000F0EE E9EBD5FFFF          <1>       jmp    sysret
                             <1>
                             <1>
                             <1> sysrmdir: ; Remove (Unlink) Directory
                             <1>       ; 29/12/2017 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>         ; INPUT ->
                             <1>         ;          EBX = Pointer to directory name
                             <1>       ; OUTPUT ->
                             <1>       ;          cf = 0 -> eax = 0
                             <1>       ;          cf = 1 -> Error code in AL
                             <1>       ;
                             <1>       ; Modified Registers: EAX (at the return of system call)
                             <1>       ;
                             <1>
0000F0F3 803D[B3030300]00    <1>       cmp    byte [u.uno], 0  ; root (super user) ?
0000F0FA 7614                <1>       jna    short sysrmdir_0
                             <1>
                             <1>       ;mov   dword [u.r0], ERR_PERM_DENIED
0000F0FC B80B000000          <1>       mov    eax, ERR_PERM_DENIED ; ERR_NOT_SUPERUSER
0000F101 A3[64030300]        <1>       mov    [u.r0], eax
0000F106 A3[C8030300]        <1>       mov    [u.error], eax
0000F10B E9AED5FFFF          <1>       jmp    error
                             <1>
                             <1> sysrmdir_0:
0000F110 89DE                <1>       mov    esi, ebx
                             <1>       ; file name is forced, change directory as temporary
                             <1>       ;mov   ax, 1
                             <1>       ;mov   [FFF_Valid], ah ; 0 ; reset
                             <1>       ;call  set_working_path
0000F112 E8E40A0000          <1>       call   set_working_path_x
0000F117 731D                <1>       jnc    short sysrmdir_1
                             <1>
0000F119 21C0                <1>       and    eax, eax  ; 0 -> Bad Path!
0000F11B 7505                <1>       jnz    short sysrmdir_err
                             <1>       ; eax = 0
                             <1> sysrmdir_not_found:
0000F11D B80C000000          <1>       mov    eax, ERR_DIR_NOT_FOUND ; Directory not found !
                             <1> sysrmdir_err:
0000F122 A3[64030300]        <1>       mov    [u.r0], eax
0000F127 A3[C8030300]        <1>       mov    [u.error], eax
0000F12C E89F0B0000          <1>       call   reset_working_path
0000F131 E988D5FFFF          <1>       jmp    error
                             <1> sysrmdir_1:
                             <1>       ;mov   esi, FindFile_Name
0000F136 66B81008            <1>       mov    ax, 0810h ; Only directories
0000F13A E82491FFFF          <1>       call   find_first_file
0000F13F 7306                <1>       jnc    short sysrmdir_2
                             <1>
                             <1>       ; eax = 2 (File not found !)
0000F141 3C02                <1>       cmp    al, 2 ; ERR_NOT_FOUND
0000F143 74D8                <1>       je     short sysrmdir_not_found
```

```
0000F145 EBDB                  <1>       jmp    short sysrmdir_err
                               <1> sysrmdir_2:
                               <1>       ; check directory attributes
                               <1>
0000F147 F6C307                <1>         test bl, 7 ; system, hidden, readonly
0000F14A 7407                  <1>         jz   short sysrmdir_3
                               <1>
0000F14C B80B000000            <1>         mov  eax, ERR_DIR_ACCESS ; 11 = 'permission denied !'
0000F151 EBCF                  <1>         jmp  short sysrmdir_err
                               <1> sysrmdir_3:
0000F153 6621D2                <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
0000F156 7407                  <1>       jz     short sysrmdir_4
                               <1>       ;mov   eax, ERR_NOT_DIR ; 'not a valid directory !'
0000F158 B813000000            <1>       mov    eax, ERR_INV_PATH_NAME ; 'bad path name !'
0000F15D EBC3                  <1>         jmp  short sysrmdir_err
                               <1> sysrmdir_4:
                               <1>       ;mov   bh, [LongName_EntryLength]
0000F15F 883D[6A630100]        <1>       mov    [DelFile_LNEL], bh ; Long name entry length (if > 0)
                               <1>       ; edi = Directory Entry Offset (DirBuff)
                               <1>       ; esi = Directory Entry (FFF Structure)
0000F165 E8CE97FFFF            <1>       call   delete_sub_directory
0000F16A 0F8372FFFFFF          <1>       jnc    sysrmdir_5
                               <1> ;     jc     short sysrmdir_6
                               <1> ;
                               <1> ;     xor    eax, eax ; 0
                               <1> ;sysrmdir_5:
                               <1> ;     mov    [u.r0], eax
                               <1> ;     ;mov   [u.error], eax
                               <1> ;     call   reset_working_path
                               <1> ;     jmp    sysret
                               <1> sysrmdir_6:
0000F170 A3[64030300]          <1>       mov    [u.r0], eax
0000F175 A3[C8030300]          <1>       mov    [u.error], eax
                               <1>
0000F17A 09C0                  <1>       or     eax, eax ; EAX = 0 -> Directory not empty!
0000F17C 741C                  <1>       jz     short sysrmdir_9
                               <1>
                               <1>       ; EAX > 0 -> Error code in AL (or AX or EAX)
                               <1>
0000F17E 833D[1E610100]01      <1>       cmp    dword [FAT_ClusterCounter], 1
0000F185 7209                  <1>       jb     short sysrmdir_8
                               <1> sysrmdir_7:
                               <1>       ; ESI = Logical DOS Drive Description Table address
0000F187 66BB00FF              <1>       mov    bx, 0FF00h ; BH = FFh -> use ESI for Drive parameters
                               <1>                  ; BL = 0 -> Recalculate free cluster count
0000F18B E834D0FFFF            <1>       call   calculate_fat_freespace
                               <1> sysrmdir_8:
0000F190 E83B0B0000            <1>       call   reset_working_path
0000F195 E924D5FFFF            <1>       jmp    error
                               <1>
                               <1> sysrmdir_9:
0000F19A A1[1E610100]          <1>       mov    eax, [FAT_ClusterCounter]
0000F19F 09C0                  <1>       or     eax, eax ; 0 ?
0000F1A1 0F847BFFFFFF          <1>       jz     sysrmdir_err
                               <1>       ; ESI = Logical DOS Drive Description Table address
0000F1A7 66BB01FF              <1>       mov    bx, 0FF01h ; BH = FFh -> use ESI for Drive parameters
                               <1>                  ; BL = 1 -> add free clusters
0000F1AB E814D0FFFF            <1>       call   calculate_fat_freespace
0000F1B0 09C9                  <1>       or     ecx, ecx
0000F1B2 74DC                  <1>         jz   short sysrmdir_8 ; ecx = 0 -> OK
                               <1>       ; ecx > 0 -> Error (Recalculation is needed)
0000F1B4 EBD1                  <1>       jmp    short sysrmdir_7
                               <1>
                               <1>
                               <1> syschdir: ; Change Current (Working) Drive & Directory (for user)
                               <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
                               <1>       ;
                               <1>         ; INPUT ->
                               <1>         ;          EBX = Directory name (ASCIIZ string) address
                               <1>       ; OUTPUT ->
                               <1>       ;           cf = 0 -> eax = 0
                               <1>       ;           cf = 1 -> Error code in AL
                               <1>       ;
                               <1>       ; Modified Registers: EAX (at the return of system call)
                               <1>       ;
                               <1>       ; NOTE: If drive name is not included, only the working
                               <1>       ; directory (for user, not for drive/OS) will be chanded.
                               <1>       ; If there is a drive name (as A:, B:, C:, D: etc.)
                               <1>       ; at the beginning of the ASCIIZ (directory) string,
                               <1>       ; working drive and working directory (for user)
                               <1>       ; will be changed together.
                               <1>       ; (When the program is terminated, MainProg -internal
                               <1>       ; shell- will reset working directory to the previous
                               <1>       ; -current- logical drive's current directory again.)
                               <1>
0000F1B6 89DE                  <1>       mov    esi, ebx
                               <1>       ; file name is not forced, change directory as temporary
0000F1B8 31C0                  <1>       xor    eax, eax
                               <1>       ;mov   [FFF_Valid], ah ; 0 ; reset
                               <1>       ;call  set_working_path
0000F1BA E8400A0000            <1>       call   set_working_path_xx
0000F1BF 731D                  <1>       jnc    short syschdir_ok
0000F1C1 21C0                  <1>       and    eax, eax  ; 0 -> Bad Path!
0000F1C3 7505                  <1>       jnz    short syschdir_err
                               <1>       ; eax = 0
                               <1> syschdir_not_found:
0000F1C5 B80C000000            <1>       mov    eax, ERR_DIR_NOT_FOUND ; Directory not found !
                               <1> syschdir_err:
0000F1CA A3[64030300]          <1>       mov    [u.r0], eax
0000F1CF A3[C8030300]          <1>       mov    [u.error], eax
0000F1D4 E8F70A0000            <1>       call   reset_working_path
0000F1D9 E9E0D4FFFF            <1>       jmp    error
                               <1> syschdir_ok:
```

```
0000F1DE 31C0                <1>       xor    eax, eax ; 0
0000F1E0 A3[64030300]        <1>       mov    [u.r0], eax
                             <1>       ;mov   [u.error], eax
0000F1E5 E9F4D4FFFF          <1>       jmp    sysret
                             <1>
                             <1>
                             <1> syschmod: ; Get & Change File (or Directory) Attributes
                             <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>         ; INPUT ->
                             <1>         ;          EBX = File/Directory (ASCIIZ) name address
                             <1>       ;         CL = New attributes (if CL < 40h)
                             <1>       ;         CL >= 40h -> Get File Attributes
                             <1>       ; OUTPUT ->
                             <1>       ;          cf = 0 -> EAX = File attributes (in AL)
                             <1>       ;          cf = 1 -> Error code in AL
                             <1>       ;
                             <1>       ; Modified Registers: EAX (at the return of system call)
                             <1>       ;
                             <1>       ; MSDOS File Attributes:    (bit value of attrib byte)
                             <1>       ;      ATTR_READ_ONLY      =      01h  (bit 0, 'R')
                             <1>       ;      ATTR_HIDDEN  =      02h  (bit 1, 'H')
                             <1>       ;      ATTR_SYSTEM  =      04h  (bit 2, 'S')
                             <1>       ;      ATTR_VOLUME_ID      =      08h  (bit 3)
                             <1>       ;      ATTR_DIRECTORY      =      10h  (bit 4)
                             <1>       ;      ATTR_ARCHIVE =      20h  (bit 5, 'A')
                             <1>       ;      ATTR_LONG_NAME      =      ATTR_READONLY |
                             <1>       ;                          ATTR_HIDDEN |
                             <1>       ;                          ATTR_SYSTEM |
                             <1>       ;                          ATTR_VOLUME_ID
                             <1>       ;      The upper two bits of attributes must be 0.
                             <1>
                             <1>       ; Note:      * If ATTR_DIRECTORY is set, only directory names
                             <1>       ;        will be searched (and S,H,R,A attributeds of
                             <1>       ;        the directory will be changed.)
                             <1>       ;      * If ATTR_VOLUME_ID is set, 'syschmod' system call
                             <1>       ;        will return with 'permission denied' error.
                             <1>       ;      * If ATTR_DIRECTORY is not set, only file names
                             <1>       ;        will be searched (and S,H,R,A attributes of the
                             <1>       ;        file will be changed.)
                             <1>       ;
                             <1>       ; (Ony Super User can change S,H,R attributes.)
                             <1>
0000F1EA 80F940              <1>       cmp    cl, 40h
0000F1ED 7327                <1>       jnb    short syschmod_0
                             <1>
0000F1EF F6C108              <1>       test   cl, 08h ; ATTR_VOLUME_ID
0000F1F2 750E                <1>       jnz    short syschmod_perm_err
                             <1>
0000F1F4 803D[B3030300]00    <1>       cmp    byte [u.uno], 0  ; root (super user) ?
0000F1FB 7619                <1>       jna    short syschmod_0
                             <1>
                             <1>       ; Not super user..
0000F1FD F6C107              <1>       test   cl, 07h      ; S,H,R attributes
0000F200 7414                <1>       jz     short syschmod_0
                             <1>
                             <1> syschmod_perm_err:
                             <1>       ;mov   dword [u.r0], ERR_PERM_DENIED
0000F202 B80B000000          <1>       mov    eax, ERR_PERM_DENIED ; 'permission denied !'
0000F207 A3[64030300]        <1>       mov    [u.r0], eax
0000F20C A3[C8030300]        <1>       mov    [u.error], eax
0000F211 E9A8D4FFFF          <1>       jmp    error
                             <1>
                             <1> syschmod_0:
0000F216 880D[B8630100]      <1>       mov    [Attributes], cl
0000F21C 89DE                <1>       mov    esi, ebx
                             <1>       ; file name is forced, change directory as temporary
                             <1>       ;mov   ax, 1
                             <1>       ;mov   [FFF_Valid], ah ; 0 ; reset
                             <1>       ;call  set_working_path
0000F21E E8D8090000          <1>       call   set_working_path_x
0000F223 731D                <1>       jnc    short syschmod_1
0000F225 21C0                <1>       and    eax, eax  ; 0 -> Bad Path!
0000F227 7505                <1>       jnz    short syschmod_err
                             <1>       ; eax = 0
                             <1> syschmod_path_not_found:
0000F229 B813000000          <1>       mov    eax, ERR_INV_PATH_NAME ; 'Bad path name !'
                             <1> syschmod_err:
0000F22E A3[64030300]        <1>       mov    [u.r0], eax
0000F233 A3[C8030300]        <1>       mov    [u.error], eax
0000F238 E8930A0000          <1>       call   reset_working_path
0000F23D E97CD4FFFF          <1>       jmp    error
                             <1> syschmod_1:
0000F242 B008                <1>       mov    al, 08h ; Except volume labels (& long names)
0000F244 A0[B8630100]        <1>       mov    al, [Attributes]
0000F249 2410                <1>       and    al, 10h ;
                             <1>       ;mov   esi, FindFile_Name
                             <1>       ;mov   ax, 1800h ; Only files
                             <1>       ;mov   ax, 0810h ; Only directories
0000F24B E81390FFFF          <1>       call   find_first_file
                             <1>       ;jnc   short syschmod_2
0000F250 72DC                <1>       jc     short syschmod_err
                             <1>
                             <1>       ;; eax = 2 (File not found !)
                             <1>       ;cmp   al, 2 ; ERR_NOT_FOUND
                             <1>       ;jne   short syschmod_err
                             <1>
                             <1>       ;and   byte [Attributes], 10h
                             <1>       ;jz    short syschmod_err
                             <1>
                             <1>       ;; Directory not found !
                             <1>       ;mov   al, 3 ; ERR_PATH_NOT_FOUND
                             <1>       ;jmp   short syschmod_err
                             <1>
                             <1> syschmod_2:
0000F252 6621D2              <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
0000F255 7407                <1>       jz     short syschmod_3
0000F257 B81A000000          <1>       mov    eax, ERR_INV_FILE_NAME ; 'invalid file name !'
0000F25C EBD0                <1>         jmp  short syschmod_err
                             <1> syschmod_3:
                             <1>       ; EDI = Directory buffer entry offset/address
                             <1>       ; BL = File (or Directory) Attributes
                             <1>       ; mov  bl, [EDI+0Bh]
                             <1>
                             <1>       ; check directory attributes
0000F25E 8A3D[B8630100]      <1>       mov    bh, [Attributes] ; new attributes
0000F264 80FF40              <1>       cmp    bh, 40h  ;>=40 -> get file/directory attributes
0000F267 732D                <1>       jnb    short syschmod_6
                             <1>
                             <1>       ; set file/directory attributes
0000F269 F6C307              <1>       test   bl, 7 ; system, hidden, readonly
0000F26C 7409                <1>         jz   short syschmod_4
                             <1>
0000F26E 803D[B3030300]00    <1>       cmp    byte [u.uno], 0  ; root (super user) ?
0000F275 778B                <1>       ja     short syschmod_perm_err
                             <1> syschmod_4:
0000F277 66817F0CA101        <1>       cmp    word [edi+DirEntry_NTRes], 01A1h ; Singlix FS
0000F27D 7424                <1>       je     short syschmod_7
                             <1>
0000F27F 887F0B              <1>       mov    [edi+0Bh], bh    ; Attributes (New!)
                             <1>
0000F282 C605[28610100]02    <1>       mov    byte [DirBuff_ValidData], 2 ; modified sign
                             <1>                                    ; to force write
0000F289 E80AB6FFFF          <1>       call   save_directory_buffer
0000F28E 729E                <1>       jc     short syschmod_err
                             <1>
                             <1> syschmod_5:
0000F290 8A1D[B8630100]      <1>       mov    bl, [Attributes]
                             <1> syschmod_6:
0000F296 0FB6C3              <1>       movzx  eax, bl
0000F299 A3[64030300]        <1>       mov    [u.r0], eax
                             <1>       ;mov   dword [u.error], 0
0000F29E E93BD4FFFF          <1>       jmp    sysret
                             <1>
                             <1> syschmod_7:
0000F2A3 29C0                <1>       sub    eax, eax
0000F2A5 8A25[26610100]      <1>         mov     ah, [DirBuff_DRV]
0000F2AB BE00010900          <1>       mov    esi, Logical_DOSDisks
0000F2B0 01C6                <1>         add     esi, eax
0000F2B2 807E04A1            <1>         cmp     byte [esi+LD_FSType], 0A1h
0000F2B6 7307                <1>       jnc    short syschmod_8
0000F2B8 B01D                <1>       mov    al, ERR_INV_DATA ; 29 = Invalid Data
0000F2BA E96FFFFFFF          <1>       jmp    syschmod_err
                             <1>
                             <1> syschmod_8:
                             <1>       ; BH = New MS-DOS File Attributes
0000F2BF 88F8                <1>       mov    al, bh ; File/Directory Attributes
0000F2C1 30E4                <1>       xor    ah, ah ; Attributes in MS-DOS format sign
0000F2C3 E8F9A0FFFF          <1>       call   change_fs_file_attributes
0000F2C8 0F8260FFFFFF        <1>       jc     syschmod_err
0000F2CE EBC0                <1>       jmp    short syschmod_5
                             <1>
                             <1>
                             <1> sysdrive: ; Get/Set Current (Working) Drive (for user)
                             <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>         ; INPUT ->
                             <1>         ;          BL = Logical DOS Drive number (0=A: ... 2=C:)
                             <1>       ;        If BL = 0FFh -> Get Current Drive
                             <1>       ; OUTPUT ->
                             <1>       ;          cf = 0 ->
                             <1>       ;               AL = Current Drive number
                             <1>       ;               AH = The Last Logical DOS Drive no.
                             <1>       ;          cf = 1 -> Error code in AL
                             <1>       ;
                             <1>       ; Modified Registers: EAX (at the return of system call)
                             <1>       ;
                             <1>       ; NOTE: If the requested logical dos drive is ready,
                             <1>       ;      it's current current directory will be the user's
                             <1>       ;      (program's) current directory.
                             <1>       ;      (When the program is terminated, MainProg -internal
                             <1>       ;      shell- will reset the previous -current- logical drive
                             <1>       ;       as current drive again).
                             <1>
0000F2D0 80FBFF              <1>       cmp    bl, 0FFh
0000F2D3 7435                <1>       je     short sysdrive_ok
0000F2D5 3A1D[D20C0100]      <1>       cmp    bl, [Last_DOS_DiskNo]
0000F2DB 771E                <1>       ja     short sysdrive_err
                             <1>
                             <1>       ; Save current drive and reset mode
                             <1>       ; for 'reset_working_path' procedure (for MainProg)
0000F2DD 30C0                <1>       xor    al, al
0000F2DF 66A3[F4650100]      <1>       mov    [SWP_Mode], ax ; ah = 0
0000F2E5 A0[FE580100]        <1>       mov    al, [Current_Drv]
0000F2EA FEC4                <1>       inc    ah ; mov ah, 1
0000F2EC 66A3[F6650100]      <1>       mov    [SWP_DRV], ax
                             <1>
0000F2F2 88DA                <1>       mov    dl, bl
0000F2F4 E8C77BFFFF          <1>       call   change_current_drive
0000F2F9 730F                <1>       jnc    short sysdrive_ok
                             <1> sysdrive_err:
0000F2FB C705[64030300]0F00- <1>       mov    dword [u.r0], ERR_DRV_NOT_RDY ; 'drive not ready !'
0000F303 0000                <1>
0000F305 E9B4D3FFFF          <1>       jmp    error
                             <1> sysdrive_ok:
0000F30A A0[FE580100]        <1>       mov    al, [Current_Drv]
```

```
0000F30F 8A25[D20C0100]       <1>       mov    ah, [Last_DOS_DiskNo]
0000F315 A3[64030300]         <1>       mov    [u.r0], eax
0000F31A E9BFD3FFFF           <1>       jmp    sysret
                              <1>
                              <1>
                              <1> sysdir: ; Get Current (Working) Drive & Directory (for user)
                              <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
                              <1>       ;
                              <1>         ; INPUT ->
                              <1>         ;          EBX = Current directory name buffer address
                              <1>       ;            (Buffer length = 92 bytes)
                              <1>       ; OUTPUT ->
                              <1>       ;          AL = Current drive (0=A: .. 2=C:)
                              <1>       ;        If CF = 1 -> AL = error code
                              <1>       ;
                              <1>       ; Modified Registers: EAX (at the return of system call)
                              <1>       ;
                              <1>       ; Note: Required directory name buffer length may be
                              <1>       ;      <= 92 bytes for current TRDOS 386 version.
                              <1>       ;      (7*12 name chars + 7 slash + 0)
                              <1>
0000F31F 89E5                 <1>       mov    ebp, esp
0000F321 83EC60               <1>       sub    esp, 96
0000F324 53                   <1>       push   ebx ; User's buffer address
0000F325 30D2                 <1>       xor    dl, dl ; 0 = current drive
0000F327 E890AAFFFF           <1>       call   get_current_directory
0000F32C 72CD                 <1>       jc     short sysdrive_err ; 'drive not ready !' error
0000F32E 89E6                 <1>       mov    esi, esp ; System's buffer address
0000F330 5F                   <1>       pop    edi  ; User's buffer address
                              <1>       ; ecx = transfer (byte) count (<=92)
0000F331 E843F4FFFF           <1>       call   transfer_to_user_buffer
0000F336 89EC                 <1>       mov    esp, ebp
0000F338 730F                 <1>       jnc    short sysdir_ok
                              <1> sysdir_err:
0000F33A C705[64030300]2E00-  <1>       mov    dword [u.r0], ERR_BUFFER  ; 'buffer error !'
0000F342 0000                 <1>
0000F344 E975D3FFFF           <1>       jmp    error
                              <1> sysdir_ok:
0000F349 8A0D[FE580100]       <1>       mov    cl, [Current_Drv]
0000F34F 890D[64030300]       <1>       mov    [u.r0], ecx
0000F355 E984D3FFFF           <1>       jmp    sysret
                              <1>
                              <1>
                              <1> sysldrvt: ; Get copy of Logical DOS Drive Description Table
                              <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
                              <1>       ;
                              <1>         ; INPUT ->
                              <1>       ;         BL = Logical DOS drive number (zero based)
                              <1>         ;          ECX = Logical DOS drv desc table buffer addr
                              <1>       ;            (Buffer length = 256 bytes)
                              <1>       ; OUTPUT ->
                              <1>       ;          cf = 0 ->
                              <1>       ;               AL = Current Drive number
                              <1>       ;               AH = The Last Logical DOS Drive no.
                              <1>       ;          cf = 1 -> Error code in AL
                              <1>       ;               AH = The Last Logical DOS Drive no.
                              <1>       ;
                              <1>       ; Modified Registers: EAX (at the return of system call)
                              <1>       ;
                              <1>       ; Note: Required description table buffer length is
                              <1>       ;      256 bytes for current TRDOS 386 version.
                              <1>
0000F35A 89CF                 <1>       mov    edi, ecx ; Destination address (user space)
0000F35C 88DC                 <1>       mov    ah, bl
0000F35E 30C0                 <1>       xor    al, al
0000F360 BE00010900           <1>       mov    esi, Logical_DOSDisks
0000F365 01C6                 <1>       add    esi, eax ; Source address (system space)
0000F367 B900010000           <1>       mov    ecx, 256 ; Byte count
                              <1>                   ; Logical Dos Drv Desc Table size
0000F36C E808F4FFFF           <1>       call   transfer_to_user_buffer
0000F371 72C7                 <1>       jc     short sysdir_err
0000F373 8A2D[D20C0100]       <1>       mov    ch, [Last_DOS_DiskNo]
0000F379 EBCE                 <1>       jmp    short sysdir_ok
                              <1>
                              <1>
                              <1> systime: ; Get System Date&Time
                              <1>       ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
                              <1>       ;
                              <1>         ; INPUT -> BL =
                              <1>       ;         0 = Get Date&Time in Unix/Epoch format
                              <1>       ;         1 = Get Time in MSDOS format
                              <1>       ;         2 = Get Date in MSDOS format
                              <1>       ;         3 = Get Date&Time in MSDOS format
                              <1>       ;         4 & other values =
                              <1>       ;            System timer ticks will be returned
                              <1>       ;            in EAX and Carry Flag will be set.
                              <1>       ;            (CF will not be set if BL = 4)
                              <1>       ; OUTPUT ->
                              <1>       ;      For BL input = 3
                              <1>       ;          EAX = Current Time (RTC)
                              <1>       ;            AL = Second (DL in MSDOS)
                              <1>       ;            AH = Minute (CL in MSDOS)
                              <1>       ;            HW of EAX = Hour (CH in MSDOS)
                              <1>       ;        EDX = Current System Date (RTC)
                              <1>       ;            DL = Day (DL in MSDOS)
                              <1>       ;            DH = Month (DH in MSDOS)
                              <1>       ;            HW of EDX = Year (CX in MSDOS)
                              <1>       ;
                              <1>       ;      For BL input = 2
                              <1>       ;        EAX = Current System Date (RTC)
                              <1>       ;            DL = Day (DL in MSDOS)
                              <1>       ;            DH = Month (DH in MSDOS)
                              <1>       ;            HW of EDX = Year (CX in MSDOS)
```

```
                                <1>     ;
                                <1>     ;       For BL input = 1
                                <1>     ;          EAX = Current Time (RTC)
                                <1>     ;            AL = Second (DL in MSDOS)
                                <1>     ;            AH = Minute (CL in MSDOS)
                                <1>     ;            HW of EAX = Hour (CH in MSDOS)
                                <1>     ;
                                <1>     ;       For BL input = 0
                                <1>     ;          EAX = Unix (Epoch) Time Ticks/Seconds
                                <1>     ;
                                <1>     ;       For BL input  = 4
                                <1>     ;          EAX = System timer ticks
                                <1>     ;
                                <1>     ;       If CF = 1 (for other values of BL input)
                                <1>     ;          EAX = System timer ticks (no error code!)
                                <1>     ;
                                <1>     ; Modified Registers: EAX, (EDX)
                                <1>     ;              (at the return of system call)
                                <1>     ;
                                <1>
0000F37B 20DB                   <1>     and     bl, bl
0000F37D 750F                   <1>     jnz     short systime_1
0000F37F E84071FFFF             <1>     call    epoch
                                <1> systime_0:
0000F384 A3[64030300]           <1>     mov     [u.r0], eax
0000F389 E950D3FFFF             <1>     jmp     sysret
                                <1> systime_1:
0000F38E 80FB04                 <1>     cmp     bl, 4
0000F391 7211                   <1>     jb      short systime_2
0000F393 A1[B8580100]           <1>     mov     eax, [TIMER_LH] ; 18.2 Hz timer ticks
                                <1>                         ; Note: [TIMER_LH] may be set
                                <1>                         ; to wrong timer value due to
                                <1>                         ; program functions.
                                <1>                         ; (This value must not be
                                <1>                         ; accepted as [TIMER_LH]/18.2
                                <1>                         ; seconds since the midnight.)
0000F398 76EA                   <1>     jna     short systime_0
0000F39A A3[64030300]           <1>     mov     [u.r0], eax
0000F39F E91AD3FFFF             <1>     jmp     error ; cf = 1 & [u.r0] = eax = timer ticks
                                <1>
                                <1> systime_2:
                                <1>     ;push   ebx
0000F3A4 E87D70FFFF             <1>     call    get_rtc_date_time
                                <1>     ;pop    ebx
0000F3A9 F6C301                 <1>     test    bl, 1
0000F3AC 7429                   <1>     jz      short systime_4
0000F3AE 30E4                   <1>     xor     ah, ah
0000F3B0 A0[30550100]           <1>     mov     al, [hour]
0000F3B5 88C2                   <1>     mov     dl, al
0000F3B7 C1E010                 <1>     shl     eax, 16
0000F3BA A0[34550100]           <1>     mov     al, [second]
0000F3BF 8A25[32550100]         <1>     mov     ah, [minute]
0000F3C5 F6C302                 <1>     test    bl, 2
0000F3C8 74BA                   <1>     jz      short systime_0
                                <1>     ; Check time & date match risk
                                <1>     ; (23:59:59 may cause to wrong
                                <1>     ; date -new day with previous date-...)
0000F3CA 80FA17                 <1>     cmp     dl, 23
0000F3CD 7206                   <1>     jb      short systime_3
0000F3CF 663D3B3B               <1>     cmp     ax, (59*256)+59 ; if hour is 23:59:59
0000F3D3 73CF                   <1>     jnb     short systime_2 ; wait for 1 second
                                <1> systime_3:
                                <1>     ; eax = time
0000F3D5 89C6                   <1>     mov     esi, eax
                                <1> systime_4:
0000F3D7 66A1[2A550100]         <1>     mov     ax, [year]
0000F3DD C1E010                 <1>     shl     eax, 16
0000F3E0 A0[2E550100]           <1>     mov     al, [day]
0000F3E5 8A25[2C550100]         <1>     mov     ah, [month]
                                <1>     ; eax = date
0000F3EB 80E301                 <1>     and     bl, 1
0000F3EE 7494                   <1>     jz      short systime_0
0000F3F0 96                     <1>     xchg    esi, eax
                                <1>     ; eax = time, esi = date
0000F3F1 8B2D[60030300]         <1>     mov     ebp, [u.usp]  ; EBP points to user's registers
                                <1>     ; (user) edx <-- (system) esi
0000F3F7 897514                 <1>     mov     [ebp+20], esi ; return to user with EDX value
0000F3FA EB88                   <1>     jmp     short systime_0
                                <1>
                                <1>
                                <1> systime: ; Set System Date&Time
                                <1>     ; 31/12/2017
                                <1>     ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
                                <1>     ;
                                <1>       ; INPUT -> BL =
                                <1>     ;           0 = Set Date&Time in Unix/Epoch format
                                <1>     ;           1 = Set Time in MSDOS format
                                <1>     ;           2 = Set Date in MSDOS format
                                <1>     ;           3 = Set Date&Time in MSDOS format
                                <1>     ;           4 = Set System Timer (Ticks)
                                <1>     ;           5 = Convert/Save current time to/as
                                <1>     ;             18.2 Hz system timer ticks
                                <1>     ;           6 = Convert MSDOS Date&Time to UNIX format
                                <1>     ;             without setting system date&time ; (test)
                                <1>     ;           7 = Convert UNIX Date&Time to MSDOS format
                                <1>     ;             without setting system date&time ; (test)
                                <1>     ;          8-0FFh = invalid !
                                <1>     ;         ECX = Time (or Timer) value in selected format
                                <1>     ;         EDX = Date value in MSDOS format if BL=2,3,6
                                <1>     ;
                                <1>     ; OUTPUT ->
                                <1>     ;       If CF = 0 ->
                                <1>     ;          EAX = Set value
```

```
                             <1>       ;     If CF = 1 -> (invalid BL input)
                             <1>       ;        EAX = Ticks count [TIMER_LH]
                             <1>       ;
                             <1>
0000F3FC 20DB                <1>       and   bl, bl ; 0
0000F3FE 7511                <1>       jnz   short sysstime_0
0000F400 89C8                <1>       mov   eax, ecx
0000F402 E84771FFFF          <1>       call  convert_from_epoch
0000F407 E8F371FFFF          <1>       call  set_rtc_date_time
0000F40C E9CDD2FFFF          <1>       jmp   sysret
                             <1> sysstime_0:
0000F411 80FB08              <1>       cmp   bl, 8
0000F414 722D                <1>       jb    short sysstime_1
                             <1>       ; invalid input (>7)
0000F416 A1[B8580100]        <1>       mov   eax, [TIMER_LH] ; 18.2 Hz timer ticks
                             <1>                             ; Note: [TIMER_LH] may be set
                             <1>                             ; to wrong timer value due to
                             <1>                             ; program functions.
                             <1>                             ; (This value must not be
                             <1>                             ; accepted as [TIMER_LH]/18.2
                             <1>                             ; seconds since the midnight.)
0000F41B A3[64030300]        <1>       mov   [u.r0], eax
0000F420 E999D2FFFF          <1>       jmp   error ; cf = 1 & [u.r0] = eax = timer ticks
                             <1>
                             <1> sysstime_8:
                             <1>       ; BL = 7
0000F425 89C8                <1>       mov   eax, ecx ; seconds since 1/1/1970 00:00:00
0000F427 E82271FFFF          <1>       call  convert_from_epoch
0000F42C 30E4                <1>       xor   ah, ah
0000F42E A0[30550100]        <1>       mov   al, [hour]
0000F433 C1E010              <1>       shl   eax, 16
0000F436 A0[34550100]        <1>       mov   al, [second]
0000F43B 8A25[32550100]      <1>       mov   ah, [minute]
0000F441 EB92                <1>       jmp   short systime_3
                             <1>
                             <1> sysstime_1:
0000F443 80FB04              <1>       cmp   bl, 4
0000F446 743F                <1>       je    short sysstime_2 ; set system timer ticks
0000F448 80FB05              <1>       cmp   bl, 5
0000F44B 754B                <1>       jne   short sysstime_4
                             <1>       ; convert current time to system timer ticks (18.2Hz)
0000F44D E8D46FFFFF          <1>       call  get_rtc_date_time
0000F452 0FB60D[30550100]    <1>       movzx ecx, byte [hour]
0000F459 B8100E0000          <1>       mov   eax, 60*60 ; 1 hour = 3600 seconds
0000F45E F7E1                <1>       mul   ecx
0000F460 89C3                <1>       mov   ebx, eax
0000F462 B13C                <1>       mov   cl, 60  ; 1 minute = 60 seconds
0000F464 0FB605[32550100]    <1>       movzx eax, byte [minute]
0000F46B F7E1                <1>       mul   ecx
0000F46D 01D8                <1>       add   eax, ebx
0000F46F 8A0D[34550100]      <1>       mov   cl, [second]
0000F475 01C8                <1>       add   eax, ecx
0000F477 B1B6                <1>       mov   cl, 182
0000F479 F7E1                <1>       mul   ecx
0000F47B 83C009              <1>       add   eax, 9
0000F47E 83D200              <1>       adc   edx, 0
0000F481 B10A                <1>       mov   cl, 10
0000F483 F7F1                <1>       div   ecx
                             <1>       ; eax = ((182*seconds)+9)/10
0000F485 89C1                <1>       mov   ecx, eax
                             <1> sysstime_2:
0000F487 890D[B8580100]      <1>       mov   [TIMER_LH], ecx ; 18.2 * seconds
                             <1> sysstime_3:
0000F48D 890D[64030300]      <1>       mov   [u.r0], ecx
0000F493 E946D2FFFF          <1>       jmp   sysret
                             <1> sysstime_4:
0000F498 80FB06              <1>       cmp   bl, 6
0000F49B 7788                <1>       ja    short sysstime_8
                             <1>
0000F49D 890D[64030300]      <1>       mov   [u.r0], ecx
                             <1>
0000F4A3 880D[34550100]      <1>       mov   [second], cl
0000F4A9 882D[32550100]      <1>       mov   [minute], ch
0000F4AF C1E910              <1>       shr   ecx, 16
0000F4B2 880D[30550100]      <1>       mov   [hour], cl
                             <1>       ; BL = 1,2,3,6
0000F4B8 80FB01              <1>       cmp   bl, 1
0000F4BB 762A                <1>       jna   short sysstime_5
                             <1>       ; BL = 2,3,6
0000F4BD 8815[2E550100]      <1>       mov   [day], dl
0000F4C3 8835[2C550100]      <1>       mov   [month], dh
0000F4C9 C1EA10              <1>       shr   edx, 16
0000F4CC 668915[2A550100]    <1>       mov   [year], dx
0000F4D3 80E303              <1>       and   bl, 3
0000F4D6 742D                <1>       jz    short sysstime_7 ; 6
                             <1>       ; BL = 2,3
0000F4D8 F6C301              <1>       test  bl, 1
0000F4DB 7419                <1>       jz    short sysstime_6 ; 2
                             <1>       ; BL = 3
0000F4DD E81D71FFFF          <1>       call  set_rtc_date_time
0000F4E2 E9F7D1FFFF          <1>       jmp   sysret
                             <1> sysstime_5:
                             <1>       ; BL = 1
0000F4E7 E85471FFFF          <1>       call  set_time_bcd
0000F4EC E82C65FFFF          <1>       call  set_rtc_time
0000F4F1 E9E8D1FFFF          <1>       jmp   sysret
                             <1> sysstime_6:
                             <1>       ; BL = 2
0000F4F6 E81871FFFF          <1>       call  set_date_bcd
0000F4FB E88C65FFFF          <1>       call  set_rtc_date
0000F500 E9D9D1FFFF          <1>       jmp   sysret
                             <1> sysstime_7:
                             <1>       ; BL = 6
                             <1>       ; [year], [month], [day],
                             <1>       ; [hour], [minute], [second]
0000F505 E8BF6FFFFF          <1>       call   convert_to_epoch
0000F50A 89C1                <1>       mov    ecx, eax ; seconds since 1/1/1970 00:00:00
0000F50C E97CFFFFFF          <1>       jmp    sysstime_3
                             <1>
                             <1> sysrename: ; Rename File (or Directory)
                             <1>       ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
                             <1>       ;
                             <1>         ; INPUT ->
                             <1>         ;          EBX = File/Directory (ASCIIZ) name address
                             <1>       ;        ECX = New name (in same dir, no path name)
                             <1>       ; OUTPUT ->
                             <1>       ;          cf = 0 -> EAX = 0
                             <1>       ;          cf = 1 -> Error code in AL
                             <1>
0000F511 803D[B3030300]00    <1>       cmp    byte [u.uno], 0  ; root (super user) ?
0000F518 7614                <1>       jna    short sysrename_0
                             <1>
                             <1> sysrename_perm_err:
                             <1>       ;mov   dword [u.r0], ERR_PERM_DENIED
0000F51A B80B000000          <1>       mov    eax, ERR_PERM_DENIED ; 'permission denied !'
0000F51F A3[64030300]        <1>       mov    [u.r0], eax
0000F524 A3[C8030300]        <1>       mov    [u.error], eax
0000F529 E990D1FFFF          <1>       jmp    error
                             <1>
                             <1> sysrename_0:
0000F52E 51                  <1>       push   ecx ; new file name address (in user space)
0000F52F 89DE                <1>       mov    esi, ebx
                             <1>       ; file name is forced, change directory as temporary
                             <1>       ;mov   ax, 1
                             <1>       ;mov   [FFF_Valid], ah ; 0 ; reset
                             <1>       ;call  set_working_path
0000F531 E8C5060000          <1>       call   set_working_path_x
0000F536 731E                <1>       jnc    short sysrename_1
0000F538 21C0                <1>       and    eax, eax  ; 0 -> Bad Path!
0000F53A 7505                <1>       jnz    short sysrename_err
                             <1>       ; eax = 0
                             <1> sysrename_path_not_found:
0000F53C B813000000          <1>       mov    eax, ERR_INV_PATH_NAME ; 'Bad path name !'
                             <1> sysrename_err:
0000F541 59                  <1>       pop    ecx ; new file name address (in user space)
                             <1> sysrename_error:
0000F542 A3[64030300]        <1>       mov    [u.r0], eax
0000F547 A3[C8030300]        <1>       mov    [u.error], eax
0000F54C E87F070000          <1>       call   reset_working_path
0000F551 E968D1FFFF          <1>       jmp    error
                             <1> sysrename_1:
0000F556 B008                <1>       mov    al, 08h ; Except volume labels (& long names)
0000F558 A0[B8630100]        <1>       mov    al, [Attributes]
0000F55D 2410                <1>       and    al, 10h ;
                             <1>       ;mov   esi, FindFile_Name
                             <1>       ;mov   ax, 1800h ; Only files
                             <1>       ;mov   ax, 0810h ; Only directories
0000F55F 66B80008            <1>       mov    ax, 0800h ; Find File or Directory
0000F563 E8FB8CFFFF          <1>       call   find_first_file
                             <1>       ;jnc   short sysrename_2
0000F568 72D7                <1>       jc     short sysrename_err
                             <1> sysrename_2:
                             <1>       ; ESI = Directory Entry (FindFile_DirEntry) Location
                             <1>       ; EDI = Directory Buffer Directory Entry Location
                             <1>       ; EAX = File Size
                             <1>       ;  BL = Attributes of The File/Directory
                             <1>       ;  BH = Long Name Yes/No Status (>0 is YES)
                             <1>       ;  DX > 0 : Ambiguous filename chars are used
                             <1>
0000F56A 6621D2              <1>       and    dx, dx ; Ambiguous filename chars used sign (DX>0)
0000F56D 7407                <1>       jz     short sysrename_3
0000F56F B81A000000          <1>       mov    eax, ERR_INV_FILE_NAME ; 'invalid file name !'
0000F574 EBCB                <1>         jmp  short sysrename_err
                             <1> sysrename_3:
                             <1>       ; EDI = Directory buffer entry offset/address
                             <1>       ; BL = File (or Directory) Attributes
                             <1>       ; mov  bl, [EDI+0Bh]
                             <1>
0000F576 5A                  <1>       pop    edx ; new file name address (in user space)
                             <1>
                             <1>       ; check file/directory attributes
0000F577 F6C307              <1>       test   bl, 7 ; system, hidden, readonly
0000F57A 759E                <1>         jnz  short sysrename_perm_err
                             <1> sysrename_4:
0000F57C 66817F0CA101        <1>       cmp    word [edi+DirEntry_NTRes], 01A1h ; Singlix FS
0000F582 7496                <1>       je     short sysrename_perm_err ; -temporary!-
                             <1>
                             <1>       ; save old file name & file info (FFF structure)
0000F584 BE[A2620100]        <1>       mov    esi, FindFile_Drv
0000F589 BF[E8630100]        <1>       mov    edi, SourceFile_Drv
0000F58E B920000000          <1>       mov    ecx, 128/4
0000F593 F3A5                <1>       rep    movsd
                             <1>
0000F595 89D6                <1>       mov    esi, edx ; new file name address (in user space)
0000F597 BF[68640100]        <1>       mov    edi, DestinationFile_Drv
0000F59C E893AEFFFF          <1>       call   parse_path_name
0000F5A1 729F                <1>       jc     short sysrename_error ; eax = 1 (Bad file name)
                             <1>
                             <1>       ; same drive ?
0000F5A3 A0[A2620100]        <1>       mov    al, [FindFile_Drv]
0000F5A8 3A05[68640100]      <1>       cmp    al, [DestinationFile_Drv]
                             <1>       ;jne   short sysrename_perm_err ; Permission denied
0000F5AE 7509                <1>       jne    short sysrename_5 ; Bad file name
                             <1>
                             <1>       ; no path name !? (rename file in same directory)
0000F5B0 803D[69640100]20    <1>       cmp    byte [DestinationFile_Directory], 20h
```

```
0000F5B7 7607                 <1>       jna    short sysrename_6
                              <1> sysrename_5:
0000F5B9 B801000000           <1>       mov    eax, ERR_BAD_CMD_ARG ; 1 = Bad file name
                              <1>                                 ; (Bad argument)
0000F5BE EB82                 <1>       jmp    short sysrename_error
                              <1> sysrename_6:
0000F5C0 803D[AA640100]20     <1>       cmp    byte [DestinationFile_Name], 20h
0000F5C7 76F0                 <1>       jna    short sysrename_5
                              <1> 
0000F5C9 BE[AA640100]         <1>       mov    esi, DestinationFile_Name
0000F5CE E84E90FFFF           <1>       call   check_filename ; is it a valid msdos file name?
0000F5D3 0F8269FFFFFF         <1>       jc     sysrename_error ; 26 = ERR_INV_FILE_NAME
                              <1> 
                              <1>       ;mov   esi, DestinationFile_Name
0000F5D9 66B80008             <1>       mov    ax, 0800h ; Find File or Directory
0000F5DD E8818CFFFF           <1>       call   find_first_file
0000F5E2 720A                 <1>       jc     short sysrename_7
                              <1> 
0000F5E4 B80E000000           <1>       mov    eax, ERR_FILE_EXISTS  ; file already exists !
0000F5E9 E954FFFFFF           <1>       jmp    sysrename_error
                              <1> sysrename_7:
                              <1>       ; eax = 2 (File not found !)
0000F5EE 3C02                 <1>       cmp    al, 2 ; ERR_NOT_FOUND
0000F5F0 0F854CFFFFFF         <1>       jne    sysrename_error
                              <1> 
                              <1>       ; 31/12/2017
                              <1>       ; Following code is also part of 'rename_file' in
                              <1>       ; 'trdosk3.s' (MainProg's 'rename' command) ; 13/11/2017
0000F5F6 BE[AA640100]         <1>       mov    esi, DestinationFile_Name ; (Rename_NewName)
0000F5FB 668B0D[62640100]     <1>       mov    cx, [SourceFile_DirEntryNumber]
0000F602 66A1[4E640100]       <1>       mov    ax, [SourceFile_DirEntry+20] ; First Cluster, HW
0000F608 C1E010               <1>       shl    eax, 16
0000F60B 66A1[54640100]       <1>       mov    ax, [SourceFile_DirEntry+26] ; First Cluster, LW
0000F611 0FB61D[37640100]     <1>       movzx  ebx, byte [SourceFile_LongNameEntryLength]
0000F618 E85FB6FFFF           <1>       call   rename_directory_entry
0000F61D 0F821FFFFFFF         <1>       jc     sysrename_error
                              <1>       ;xor   eax, eax
0000F623 A3[64030300]         <1>       mov    [u.r0], eax ; 0
                              <1>       ;mov   [u.error], eax
0000F628 E8A3060000           <1>       call   reset_working_path
0000F62D E9ACD0FFFF           <1>       jmp    sysret
                              <1> 
                              <1> sysmem: ; Get Total&Free Memory amount
                              <1>       ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
                              <1>       ;
                              <1>         ; INPUT ->
                              <1>       ;      none
                              <1>       ; OUTPUT ->
                              <1>       ;      EAX = Total memory count (in bytes)
                              <1>       ;      EBX = Virtually available memory amount (in bytes)
                              <1>       ;            = 4GB - CORE (4MB)
                              <1>       ;      ECX = Free memory count (in bytes)
                              <1>       ;      EDX = Calculated free memory count (in bytes)
                              <1> 
0000F632 A1[3C580100]         <1>       mov    eax, [memory_size] ; in pages
0000F637 C1E00C               <1>       shl    eax, 12                ; in bytes
0000F63A A3[64030300]         <1>       mov    [u.r0], eax
0000F63F E8043DFFFF           <1>       call   calc_free_mem
                              <1>       ; edx = calculated free pages
                              <1>       ; ecx = 0
0000F644 8B2D[60030300]       <1>       mov    ebp, [u.usp]  ; EBP points to user's registers
0000F64A C745100000C0FF       <1>       mov    dword [ebp+16], ECORE ; EBX (for user)
                              <1>                            ; 0FFC00000h ; 4GB - 4MB
0000F651 C1E20C               <1>       shl    edx, 12
0000F654 895514               <1>       mov    [ebp+20], edx ; EDX (for user)
0000F657 8B0D[40580100]       <1>       mov    ecx, [free_pages]
0000F65D C1E10C               <1>       shl    ecx, 12      ; free bytes
0000F660 894D18               <1>       mov    [ebp+24], ecx ; ECX (for user)
                              <1>       ;mov   [free_pages], edx
0000F663 E976D0FFFF           <1>       jmp    sysret
                              <1> 
                              <1> sysprompt:
                              <1>       ; Set TRDOS 386 Command Interpreter (MainProg) prompt
                              <1>       ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
                              <1>       ;
                              <1>         ; INPUT ->
                              <1>       ;      EBX = 0 -> use default prompt
                              <1>       ;      EBX > 0 -> prompt string (ASCIIZ) address
                              <1>       ;              (Max. 11 characters except ZERO tail)
                              <1>       ; OUTPUT ->
                              <1>       ;      (EAX = 0)
                              <1>       ;      CF = 0 -> Successful
                              <1>       ;      CF = 1 -> Failed
                              <1> 
0000F668 21DB                 <1>       and    ebx, ebx
0000F66A 750A                 <1>       jnz    short sysprompt_0
                              <1> 
0000F66C E8F685FFFF           <1>       call   default_command_prompt ; '['+'TRDOS'+']'
0000F671 E968D0FFFF           <1>       jmp    sysret
                              <1> 
                              <1> sysprompt_0:
0000F676 31C0                 <1>       xor    eax, eax
0000F678 A3[64030300]         <1>       mov    [u.r0], eax
0000F67D 89DE                 <1>       mov    esi, ebx
0000F67F B90C000000           <1>       mov    ecx, 12
0000F684 89E5                 <1>       mov    ebp, esp
0000F686 29CC                 <1>       sub    esp, ecx
0000F688 49                   <1>       dec    ecx ; 11
0000F689 89E7                 <1>       mov    edi, esp
0000F68B E833F1FFFF           <1>       call   transfer_from_user_buffer
0000F690 7211                 <1>       jc     short sysprompt_err
0000F692 803E20               <1>       cmp    byte [esi], 20h
0000F695 760C                 <1>       jna    short sysprompt_err
```

```
0000F697 E8DD85FFFF           <1>       call   set_command_prompt
0000F69C 89EC                 <1>       mov    esp, ebp
0000F69E E93BD0FFFF           <1>       jmp    sysret
                              <1> sysprompt_err:
                              <1> syspath_err:
0000F6A3 89EC                 <1>       mov    esp, ebp
0000F6A5 E914D0FFFF           <1>       jmp    error
                              <1>
                              <1> syspath:
                              <1>       ; Get/Set Run Path
                              <1>       ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
                              <1>       ;
                              <1>         ; INPUT ->
                              <1>       ;      EBX = 0 -> get path (to buffer address in ECX)
                              <1>       ;      EBX > 0 -> set path
                              <1>       ;          EBX = Path string buffer address (ASCIIZ)
                              <1>       ;               (Path description except 'PATH=')
                              <1>       ;      ECX = Buffer address (if EBX = 0)
                              <1>       ;            (ECX will not be used if EBX > 0)
                              <1>       ;      DL = Buffer size (0 = 256 byte)
                              <1>       ;
                              <1>       ; OUTPUT ->
                              <1>       ;      CF = 0 -> Successful (EAX = String length)
                              <1>       ;      CF = 1 -> Failed (EAX = 0)
                              <1>       ;
                              <1>       ; NOTE: 'PATH=' or 'PATH' must be excluded
                              <1>       ;  (It must not be at the beginning of the string.)
                              <1>
0000F6AA 89E5                 <1>       mov    ebp, esp
0000F6AC 81EC00010000         <1>       sub    esp, 256
0000F6B2 89E7                 <1>       mov    edi, esp
                              <1>
0000F6B4 31C0                 <1>       xor    eax, eax
0000F6B6 A3[64030300]         <1>       mov    [u.r0], eax
                              <1>
0000F6BB 21DB                 <1>       and    ebx, ebx
0000F6BD 752E                 <1>       jnz    short syspath_0
                              <1>
                              <1>       ; EBX = 0 -> get run path
0000F6BF 89CB                 <1>       mov    ebx, ecx ; buffer addr (in user's mem space)
0000F6C1 BE[9F0D0100]         <1>       mov    esi, Cmd_Path  ; 'PATH' address
0000F6C6 0FB6CA               <1>       movzx  ecx, dl
0000F6C9 80E901               <1>       sub    cl, 1 ; 0 -> 255, 1 -> 0
0000F6CC 6683D101             <1>       adc    cx, 1 ; 255 -> 256, 0 -> 1
                              <1>       ; EDI = Output buffer
                              <1>       ; CX = Buffer length
                              <1>       ; AL = 0 -> use ASCIIZ word in [ESI]
                              <1>       ; ESI = 'PATH' address (with zero tail)
0000F6D0 E8D89DFFFF           <1>       call   get_environment_string
0000F6D5 72CC                 <1>       jc     short syspath_err
0000F6D7 89DF                 <1>       mov    edi, ebx ; User's buffer address
0000F6D9 89E6                 <1>       mov    esi, esp
                              <1>       ; EDI = User's buffer address
                              <1>       ; ECX = transfer (byte) count
0000F6DB E899F0FFFF           <1>       call   transfer_to_user_buffer
0000F6E0 72C1                 <1>       jc     short syspath_err
0000F6E2 890D[64030300]       <1>       mov    [u.r0], ecx
0000F6E8 E9F1CFFFFF           <1>       jmp    sysret
                              <1>
                              <1> syspath_0:
0000F6ED 89DE                 <1>       mov    esi, ebx
0000F6EF 0FB6CA               <1>       movzx  ecx, dl
0000F6F2 80E901               <1>       sub    cl, 1 ; 0 -> 255, 1 -> 0
0000F6F5 6683D101             <1>       adc    cx, 1 ; 255 -> 256, 0 -> 1
0000F6F9 E8C5F0FFFF           <1>       call   transfer_from_user_buffer
0000F6FE 72A3                 <1>       jc     short syspath_err
                              <1>       ;(*) 'PATH=' will be added to
                              <1>       ;          the head of the string
0000F700 83EC08               <1>       sub    esp, 8 ;(*)
0000F703 89FE                 <1>       mov    esi, edi ;(*)
0000F705 E8879DFFFF           <1>       call   set_path_x ;(*)
0000F70A 7297                 <1>       jc     short syspath_err
0000F70C 8915[64030300]       <1>       mov    [u.r0], edx ; run path string length
0000F712 E9C7CFFFFF           <1>       jmp    sysret
                              <1>
                              <1> sysenv:
                              <1>       ; Get/Set Environment Variables
                              <1>       ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
                              <1>       ;
                              <1>         ; INPUT ->
                              <1>       ;      EBX = 0 -> get (all) environment variables
                              <1>       ;            (Required Buffer length = 512 bytes)
                              <1>       ;      EBX > 0 -> set (one) environment variable
                              <1>       ;            (If there is not a '=' after
                              <1>       ;            the environment variable name, it will
                              <1>       ;            accepted as 'get environment variable'.)
                              <1>       ;             EBX = Buffer address
                              <1>       ;      ECX = Buffer address (if EBX = 0)
                              <1>       ;            (ECX will not be used if EBX > 0)
                              <1>       ;            (Note: Buffer size is 512 bytes.)
                              <1>       ;      DL = Buffer size (0 = 256 byte)
                              <1>       ;           (For one envrionment variable)
                              <1>       ;
                              <1>       ; OUTPUT ->
                              <1>       ;      (EAX = 0)
                              <1>       ;      CF = 0 -> Successful (EAX = String length)
                              <1>       ;      CF = 1 -> Failed (EAX = 0)
                              <1>       ;
                              <1>       ; Note: Environment variable name, for example,
                              <1>       ;      'PATH=' must be included at the beginning
                              <1>       ;      of the environment string. If the variable
                              <1>       ;      name is as 'PATH' but it is not as 'PATH='
                              <1>       ;      the variable string (row) will be returned.
```

```
                             <1>       ;      If variable name is as 'PATH=' but there is
                             <1>       ;      not a following text after the variable name,
                             <1>       ;      the environment variable will be reset/deleted.
                             <1>
0000F717 89E5                <1>       mov    ebp, esp
0000F719 81EC00020000        <1>       sub    esp, 512
0000F71F 89E7                <1>       mov    edi, esp
                             <1>
0000F721 31C0                <1>       xor    eax, eax
0000F723 A3[64030300]        <1>       mov    [u.r0], eax
                             <1>
0000F728 21DB                <1>       and    ebx, ebx
0000F72A 7524                <1>       jnz    short sysenv_0
                             <1>
                             <1>       ; EBX = 0 -> get (all) environment variables
0000F72C 89EC                <1>       mov    esp, ebp
0000F72E BE00300900          <1>       mov    esi, Env_Page  ; Environment page
0000F733 89CF                <1>       mov    edi, ecx ; buffer addr (in user's mem space)
0000F735 B900020000          <1>       mov    ecx, 512
0000F73A E83AF0FFFF          <1>       call   transfer_to_user_buffer
0000F73F 0F8279CFFFFF        <1>       jc     error
0000F745 890D[64030300]      <1>       mov    [u.r0], ecx
0000F74B E98ECFFFFF          <1>       jmp    sysret
                             <1>
                             <1> sysenv_0:
0000F750 89DE                <1>       mov    esi, ebx ; * ; user's buffer address
0000F752 0FB6CA              <1>       movzx  ecx, dl
0000F755 80E901              <1>       sub    cl, 1 ; 0 -> 255, 1 -> 0
0000F758 6683D101            <1>       adc    cx, 1 ; 255 -> 256, 0 -> 1
0000F75C E862F0FFFF          <1>       call   transfer_from_user_buffer
0000F761 723F                <1>       jc     short sysenv_err
0000F763 89FE                <1>       mov    esi, edi
0000F765 8A06                <1>       mov    al, [esi]
0000F767 3C20                <1>       cmp    al, 20h
0000F769 7637                <1>       jna    short sysenv_err
0000F76B 3C3D                <1>       cmp    al, '='
0000F76D 7433                <1>       je     short sysenv_err
0000F76F 56                  <1>       push   esi
                             <1> sysenv_1:
0000F770 46                  <1>       inc    esi
0000F771 803E3D              <1>       cmp    byte [esi], '='
0000F774 7433                <1>       je     short sysenv_3
0000F776 803E20              <1>       cmp    byte [esi], 20h
0000F779 73F5                <1>       jnb    short sysenv_1
0000F77B C60600              <1>       mov    byte [esi], 0
0000F77E 5E                  <1>       pop    esi
                             <1>       ; EDI = Output buffer
                             <1>       ; CX = Buffer length
0000F77F 30C0                <1>       xor    al, al
                             <1>       ; AL = 0 -> use ASCIIZ word in [ESI]
                             <1>       ; ESI = Environment variable name address
0000F781 E8279DFFFF          <1>       call   get_environment_string
0000F786 721A                <1>       jc     short sysenv_err
0000F788 89DF                <1>       mov    edi, ebx ; * ; user's buffer address
0000F78A 89C1                <1>       mov    ecx, eax ; String length
0000F78C 89E6                <1>       mov    esi, esp
                             <1>       ; ESI = system buffer address
                             <1>       ; EDI = User's buffer address
                             <1>       ; ECX = transfer (byte) count
0000F78E E8E6EFFFFF          <1>       call   transfer_to_user_buffer
0000F793 720D                <1>       jc     short sysenv_err
0000F795 890D[64030300]      <1>       mov    [u.r0], ecx ; transfer (byte) count
                             <1> sysenv_2:
0000F79B 89EC                <1>       mov    esp, ebp
0000F79D E93CCFFFFF          <1>       jmp    sysret
                             <1> sysenv_err:
0000F7A2 89EC                <1>       mov    esp, ebp
0000F7A4 E915CFFFFF          <1>       jmp    error
                             <1> sysenv_3:
0000F7A9 46                  <1>       inc    esi
0000F7AA 803E20              <1>       cmp    byte [esi], 20h
0000F7AD 73FA                <1>       jnb    short sysenv_3
0000F7AF C60600              <1>       mov    byte [esi], 0
0000F7B2 5E                  <1>       pop    esi
0000F7B3 E8B89DFFFF          <1>       call   set_environment_string
0000F7B8 72E8                <1>       jc     short sysenv_err
0000F7BA 8915[64030300]      <1>       mov    [u.r0], edx
0000F7C0 EBD9                <1>       jmp    short sysenv_2
                             <1>
                             <1>
                             <1> ; temporary - 24/01/2016
                             <1>
                             <1> iget:
0000F7C2 C3                  <1>       retn
                             <1> isintr:
0000F7C3 C3                  <1>       retn
                             <1> iopen:
0000F7C4 C3                  <1>       retn
                             <1> iclose:
0000F7C5 C3                  <1>       retn
                             <1> sndc:
0000F7C6 C3                  <1>       retn
                             <1> access:
0000F7C7 C3                  <1>       retn
                             <1> sleep:
0000F7C8 C3                  <1>       retn
                                 %include 'trdosk7.s' ; 24/01/2016
                             <1> ; ****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - DISK READ&WRITE : trdosk7.s
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 25/02/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 24/01/2016
```

```
                              <1> ; ----------------------------------------------------------------------------
                              <1> ; Assembler: NASM version 2.11 (trdos386.s)
                              <1> ; ----------------------------------------------------------------------------
                              <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
                              <1> ; DISK_IO.ASM (20/07/2011)
                              <1> ; ****************************************************************************
                              <1> ; DISK_IO.ASM (c) 2009-2011 Erdogan TAN [ 04/07/2009 ] Last Update: 20/07/2011
                              <1> 
                              <1> disk_write:
                              <1>       ; 25/02/2016
                              <1>       ; 24/02/2016
                              <1>       ; 23/02/2016
0000F7C9 807E0500             <1>       cmp    byte [esi+LD_LBAYes], 0
0000F7CD 777B                 <1>         ja      short lba_write
                              <1> 
                              <1> chs_write:
                              <1>       ; 25/02/2016
                              <1>       ; 23/02/2016
0000F7CF C605[F1610100]03     <1>       mov    byte [disk_rw_op], 3 ; CHS write
0000F7D6 EB0D                 <1>       jmp    short chs_rw
                              <1> 
                              <1> disk_read:
                              <1>       ; 25/02/2016
                              <1>       ; 24/02/2016
                              <1>       ; 23/02/2016
                              <1>       ; 17/02/2016
                              <1>       ; 14/02/2016
                              <1>       ; 31/01/2016 (TRDOS 386 =  TRDOS v2.0)
                              <1>       ; 17/10/2010
                              <1>       ; 18/04/2010
                              <1>       ;
                              <1>       ; INPUT -> EAX = Logical Block Address
                              <1>       ;         ESI = Logical Dos Disk Table Offset (DRV)
                              <1>       ;         ECX = Sector Count
                              <1>       ;         EBX = Destination Buffer
                              <1>       ; OUTPUT ->
                              <1>       ;         cf = 0 or cf = 1
                              <1>       ; (Modified registers: EAX, EBX, ECX, EDX)
                              <1> 
0000F7D8 807E0500             <1>       cmp    byte [esi+LD_LBAYes], 0
0000F7DC 7775                 <1>         ja      short lba_read
                              <1> 
                              <1> chs_read:
                              <1>       ; 25/02/2016
                              <1>       ; 24/02/2016
                              <1>       ; 23/02/2016
                              <1>       ; 31/01/2016 (TRDOS 386 =  TRDOS v2.0)
                              <1>       ; 20/07/2011
                              <1>       ; 04/07/2009
                              <1>       ;
                              <1>       ; INPUT -> EAX = Logical Block Address
                              <1>       ;         ECX = Number of sectors to read
                              <1>       ;         ESI = Logical Dos Disk Table Offset (DRV)
                              <1>       ;         EBX = Destination Buffer
                              <1>       ; OUTPUT ->
                              <1>       ;         cf = 0 or cf = 1
                              <1>       ; (Modified registers: EAX; EBX, ECX, EDX)
                              <1> 
                              <1>       ; 23/02/2016
0000F7DE C605[F1610100]02     <1>       mov    byte [disk_rw_op], 2 ; CHS read
                              <1> 
                              <1> chs_rw:
                              <1>       ;;movzx      edx, word [esi+LD_BPB+SecPerTrack]
                              <1>       ;movzx edx, byte [esi+LD_BPB+SecPerTrack] ; <= 63
                              <1>       ;mov   [disk_rw_spt], dl
                              <1> 
                              <1> chs_read_next_sector:
0000F7E5 C605[F2610100]04     <1>       mov    byte [retry_count], 4
                              <1> 
                              <1> chs_read_retry:
                              <1>       ;mov   [sector_count], ecx ; 23/02/2016
                              <1> 
0000F7EC 50                   <1>       push   eax                 ; Linear sector #
0000F7ED 51                   <1>       push   ecx                 ; # of FAT/FILE/DIR sectors
                              <1> 
0000F7EE 0FB74E1E             <1>       movzx  ecx, word [esi+LD_BPB+SecPerTrack]
                              <1>       ;movzx ecx, byte [disk_rw_spt] ; 23/02/2016
0000F7F2 29D2                 <1>       sub    edx, edx
0000F7F4 F7F1                 <1>       div    ecx
                              <1>       ; eax = track, dx (dl ) = sector (on track)
                              <1>       ;sub   cl, dl ; 24/02/2016 (spt - sec)
                              <1>       ;push  ecx ; *
0000F7F6 6689D1               <1>       mov    cx, dx              ; Sector (zero based)
0000F7F9 6641                 <1>       inc    cx                  ; To make it 1 based
0000F7FB 6651                 <1>       push   cx
0000F7FD 668B4E20             <1>       mov    cx, [esi+LD_BPB+Heads]
0000F801 6629D2               <1>       sub    dx, dx
0000F804 F7F1                 <1>       div    ecx                 ; Convert track to head & cyl
                              <1>       ; eax (ax) = cylinder, dx (dl) = head (max. FFh)
0000F806 88D6                 <1>       mov    dh, dl
0000F808 6659                 <1>       pop    cx                  ; AX=Cyl, DH=Head, CX=Sector
0000F80A 8A5602               <1>       mov    dl, [esi+LD_PhyDrvNo]
                              <1> 
0000F80D 88C5                 <1>       mov    ch, al              ; NOTE: max. 1023 cylinders !
0000F80F C0CC02               <1>       ror    ah, 2               ; Rotate 2 bits right
0000F812 08E1                 <1>       or     cl, ah
                              <1> 
                              <1>       ; 24/02/2016
                              <1>       ;pop   eax ; * (spt - sec) (example: 63 - 0 = 63)
                              <1>       ;cmp   eax, [sector_count]
                              <1>       ;jb    short chs_write_sectors
                              <1>       ;je    short chs_read_sectors
                              <1>       ;; (# of sectors to read is more than remaining sectors on the track)
```

```
                             <1>       ;mov   al, [sector_count]
                             <1> ;chs_read_sectors: ; read or write !
0000F814 B001                <1>       mov    al, 1 ; 25/02/2016
0000F816 8A25[F1610100]      <1>       mov    ah, [disk_rw_op]  ; 02h = chs read, 03h = chs write
                             <1>       ;
0000F81C E8E549FFFF          <1>       call   int13h             ; BIOS Service func ( ah ) = 2
                             <1>                                         ; Read disk sectors
                             <1>                                         ; AL-sec num CH-track CL-sec
                             <1>                                         ; DH-head DL-drive ES:BX-buffer
                             <1>                                         ; CF-flag AH-stat AL-sec read
                             <1>                                       ; If CF = 1 then (If AH > 0)
0000F821 8825[F3610100]      <1>       mov    [disk_rw_err], ah
                             <1>
0000F827 59                  <1>       pop    ecx
0000F828 58                  <1>       pop    eax
0000F829 7314                <1>       jnc    short chs_read_ok
                             <1>
0000F82B 803D[F3610100]09    <1>       cmp    byte [disk_rw_err], 09h ; DMA crossed 64K segment boundary
0000F832 7408                <1>       je     short chs_read_error_retn
                             <1>
0000F834 FE0D[F2610100]      <1>       dec    byte [retry_count]
0000F83A 75B0                <1>       jnz    short chs_read_retry
                             <1>
                             <1> chs_read_error_retn:
0000F83C F9                  <1>       stc
                             <1>       ;retn
0000F83D EB69                <1>       jmp    short update_drv_error_byte
                             <1>
                             <1> ;chs_write_sectors: ; read or write
                             <1>       ;; (# of sectors to read is less than remaining sectors on the track)
                             <1>       ;mov   [sector_count], al
                             <1>       ;jmp   short chs_read_sectors
                             <1>
                             <1> chs_read_ok:
                             <1>       ;; 23/02/2016
                             <1>       ;movzx edx, byte [sector_count] ; sector count (<= spt)
                             <1>         ;sub    ecx, edx  ; remaining sector count
                             <1>       ;jna   short update_drv_error_byte
                             <1>       ;add   eax, edx ; next disk sector
                             <1>       ;shl   edx, 9 ; 512 * sector count
                             <1>       ;add   ebx, edx ; next buffer byte address
                             <1>         ;jmp     chs_read_next_sector
                             <1>       ; 25/02/2016
0000F83F 40                  <1>       inc    eax ; next sector
0000F840 81C300020000        <1>       add    ebx, 512
0000F846 E29D                <1>       loop   chs_read_next_sector
0000F848 EB5E                <1>       jmp    short update_drv_error_byte
                             <1>
                             <1> lba_write:
                             <1>       ; 23/02/2016
0000F84A C605[F1610100]1C    <1>       mov    byte [disk_rw_op], 1Ch ; LBA write
0000F851 EB07                <1>       jmp    short lba_rw
                             <1>
                             <1> lba_read:
                             <1>       ; 23/02/2016
                             <1>       ; 17/02/2016
                             <1>       ; 14/02/2016
                             <1>       ; 13/02/2016
                             <1>       ; 31/01/2016 (TRDOS 386 =  TRDOS v2.0)
                             <1>       ; 10/07/2015 (Retro UNIX 386 v1)
                             <1>       ;
                             <1>       ; INPUT -> EAX = Logical Block Address
                             <1>       ;         ESI = Logical Dos Disk Table Offset (DRV)
                             <1>       ;         ECX = Sector Count
                             <1>       ;         EBX = Destination Buffer
                             <1>       ; OUTPUT ->
                             <1>       ;         cf = 0 or cf = 1
                             <1>       ; (Modified registers: EAX, EBX, ECX, EDX)
                             <1>
                             <1>       ; LBA read/write (with private LBA function)
                             <1>       ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))
                             <1>
                             <1>
                             <1>       ; 23/02/2016
0000F853 C605[F1610100]1B    <1>       mov    byte [disk_rw_op], 1Bh ; LBA read
                             <1>
                             <1> lba_rw:
                             <1>       ; 17/02/2016
0000F85A 57                  <1>       push   edi
                             <1>
0000F85B 890D[F4610100]      <1>       mov    [sector_count], ecx ; total sector (read) count
                             <1>
0000F861 8A5602              <1>       mov    dl, [esi+LD_PhyDrvNo]
                             <1>       ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
                             <1>
                             <1> lba_read_next:
0000F864 81F900010000        <1>       cmp    ecx, 256
0000F86A 7605                <1>       jna    short lba_read_rsc
0000F86C B900010000          <1>       mov    ecx, 256 ; 17/02/2016
                             <1> lba_read_rsc:
0000F871 290D[F4610100]      <1>       sub    [sector_count], ecx ; remain sectors
                             <1>
0000F877 89CF                <1>       mov    edi, ecx
0000F879 89C1                <1>       mov    ecx, eax ; sector number/address
                             <1>
0000F87B C605[F2610100]04    <1>       mov    byte [retry_count], 4
                             <1> lba_read_retry:
0000F882 89F8                <1>       mov    eax, edi
                             <1>       ;
                             <1>       ; ecx = sector number
                             <1>       ; al = sector count (0 - 255) /// (0 = 256)
                             <1>       ; dl = drive number
                             <1>       ; ebx = buffer offset
```

```
                                  <1>       ;
                                  <1>       ; Function 1Bh = LBA read, 1Ch = LBA write
                                  <1>       ; 23/02/2016
0000F884 8A25[F1610100]           <1>       mov    ah, [disk_rw_op] ; 1Bh = LBA read, 1Ch = LBA write
0000F88A E87749FFFF               <1>       call   int13h
                                  <1>       ; al = ? (changed)
                                  <1>       ; ah = error code
0000F88F 8825[F3610100]           <1>       mov    [disk_rw_err], ah
0000F895 7334                     <1>       jnc    short lba_read_ok
0000F897 80FC80                   <1>       cmp    ah, 80h ; time out?
0000F89A 740A                     <1>         je       short lba_read_stc_retn
0000F89C FE0D[F2610100]           <1>       dec    byte [retry_count]
0000F8A2 7FDE                     <1>       jg     short lba_read_retry
0000F8A4 743A                     <1>       jz     short lba_read_reset
                                  <1>       ; sf =  1
                                  <1>
                                  <1> lba_read_stc_retn:
0000F8A6 F9                       <1>       stc
                                  <1> lba_read_retn:
0000F8A7 5F                       <1>       pop    edi
                                  <1>
                                  <1> update_drv_error_byte:
0000F8A8 9C                       <1>       pushf
0000F8A9 53                       <1>       push   ebx
0000F8AA 6651                     <1>       push   cx
                                  <1>       ;or    ecx, ecx
                                  <1>       ;jz    short udrv_errb0
0000F8AC 8A0D[F3610100]           <1>       mov    cl, [disk_rw_err]
                                  <1> udrv_errb0:
0000F8B2 0FB65E02                 <1>       movzx  ebx, byte [esi+LD_PhyDrvNo]
0000F8B6 80FB02                   <1>       cmp    bl, 2
0000F8B9 7203                     <1>         jb       short udrv_errb1
0000F8BB 80EB7E                   <1>       sub    bl, 7Eh
                                  <1>       ;cmp   bl, 5
                                  <1>       ;ja    short udrv_errb2
                                  <1> udrv_errb1:
0000F8BE 81C3[495D0000]           <1>         add      ebx, drv.error ; 13/02/2016
0000F8C4 880B                     <1>       mov    [ebx], cl ; error code
                                  <1> udrv_errb2:
0000F8C6 6659                     <1>       pop    cx
0000F8C8 5B                       <1>       pop    ebx
0000F8C9 9D                       <1>       popf
0000F8CA C3                       <1>       retn
                                  <1>
                                  <1> lba_read_ok:
0000F8CB 89C8                     <1>       mov    eax, ecx ; sector number
0000F8CD 01F8                     <1>       add    eax, edi ; sector number (next)
0000F8CF C1E709                   <1>       shl    edi, 9 ; sector count * 512
0000F8D2 01FB                     <1>       add    ebx, edi ; next buffer offset
                                  <1>
0000F8D4 8B0D[F4610100]           <1>       mov    ecx, [sector_count] ; remaining sectors
0000F8DA 09C9                     <1>       or     ecx, ecx
0000F8DC 7586                     <1>       jnz    short lba_read_next
0000F8DE EBC7                     <1>       jmp    short lba_read_retn
                                  <1>
                                  <1> lba_read_reset:
0000F8E0 B40D                     <1>       mov    ah, 0Dh ; Alternate reset
0000F8E2 E81F49FFFF               <1>         call int13h
                                  <1>       ; al = ? (changed)
                                  <1>       ; ah = error code
0000F8E7 7399                     <1>       jnc    short lba_read_retry
0000F8E9 EBBC                     <1>       jmp    short lba_read_retn
                                      %include 'trdosk8.s' ; 24/01/2016
                                  <1> ; ****************************************************************************
                                  <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - MAIN PROGRAM : trdosk8.s
                                  <1> ; ----------------------------------------------------------------------------
                                  <1> ; Last Update: 30/12/2017
                                  <1> ; ----------------------------------------------------------------------------
                                  <1> ; Beginning: 24/01/2016
                                  <1> ; ----------------------------------------------------------------------------
                                  <1> ; Assembler: NASM version 2.11 (trdos386.s)
                                  <1> ; ----------------------------------------------------------------------------
                                  <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                                  <1> ; u0.s (20/11/2015), u4.s (14/10/2015)
                                  <1> ; ****************************************************************************
                                  <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
                                  <1> ; TRDOS2.ASM (09/11/2011)
                                  <1> ; ----------------------------------------------------------------------------
                                  <1> ; DIR.ASM (c) 2004-2011 Erdogan TAN  [07/01/2004] Last Update: 09/10/2011
                                  <1>
                                  <1> set_run_sequence:
                                  <1>       ; 23/12/2016
                                  <1>       ; 10/06/2016
                                  <1>       ; 22/05/2016
                                  <1>       ; 20/05/2016
                                  <1>       ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
                                  <1>       ; TRDOS 386 feature only !
                                  <1>       ;
                                  <1>       ; INPUT ->
                                  <1>       ;      AL = process number (next process)
                                  <1>       ;
                                  <1>       ;      this process must be added to run sequence
                                  <1>       ;
                                  <1>       ;      [u.pri] = priority of present process
                                  <1>       ;
                                  <1>       ;      DL = priority (queue)
                                  <1>       ;           0 = background (low) ; run on background
                                  <1>       ;           1 = regular (normal) ; run as regular
                                  <1>       ;           2 = event (high)     ; run for event
                                  <1>       ;
                                  <1>       ;      1) If the requested process is already running:
                                  <1>       ;         a) If present priority is high ([u.pri]=2)
                                  <1>       ;            and requested priority is also high,
                                  <1>       ;           there is nothing to do! Because it has been
                                  <1>       ;           done already (before this attempt).
                                  <1>       ;        b) If present priority is high ([u.pri]=2)
                                  <1>       ;           and requested priority is not high, there is
                                  <1>       ;           nothing to do! Because, it's current
                                  <1>       ;           run queue is unspecified, here. (It may be in
                                  <1>       ;           a waiting list or in a run queue; if the new
                                  <1>       ;           priority would be used to add it to relavant
                                  <1>         ;             run queue, this would be wrong, unnecessary
                                  <1>       ;           and destabilizing duplication!)
                                  <1>       ;        c) If present priority is not high ([u.pri]<2)
                                  <1>         ;             and requested priority is high (event),
                                  <1>       ;           process will be added to present priority's
                                  <1>       ;           run queue and then, priority will be changed
                                  <1>       ;           to high ([u.pri]=2).
                                  <1>       ;        d) If present priority is not high ([u.pri]<2)
                                  <1>       ;           and requested priority is not high, [u.pri]
                                  <1>       ;           value will be changed. There is nothing to do
                                  <1>       ;           in addition. (The new priority value will be
                                  <1>       ;           used by 'tswap/tswitch' procedure at 'sysret'
                                  <1>       ;           or 'sysrele' stage.)
                                  <1>       ;
                                  <1>       ;     2) If the requested process is not running:
                                  <1>       ;        a) If requested priority of the requested
                                  <1>       ;           (next) process is high (event) and priority
                                  <1>       ;           of present process is not high, the requested
                                  <1>       ;           process will be added to ('runq_event') high
                                  <1>       ;           priority run queue and then present (running)
                                  <1>       ;           process will be stopped (swapped/switched out)
                                  <1>       ;           immediately if it is in user mode, or it's
                                  <1>       ;           [u.quant] value will be reset to 0 and (then)
                                  <1>       ;           it will be stopped at 'sysret' stage.
                                  <1>       ;        b) If requested priority of the requested
                                  <1>       ;           (next) process is high (event) and priority
                                  <1>       ;           of present process is also high, the requested
                                  <1>       ;           process will be added to ('runq_event') high
                                  <1>       ;           priority run queue and present (running)
                                  <1>       ;           process will be allowed to run until it's
                                  <1>       ;           time quantum will be elapsed ([u.quant]=0).
                                  <1>       ;        c) If requested priority of the requested
                                  <1>       ;           (next) process is not high ('run for event'),
                                  <1>       ;           there is nothing to do. Because, it's current
                                  <1>       ;           run queue is unspecified, here. (It may be in
                                  <1>       ;           a waiting list or in a run queue; if the new
                                  <1>       ;           priority would be used to add it to relavant
                                  <1>         ;             run queue, this would be wrong, unnecessary
                                  <1>       ;           and destabilizing duplication!)
                                  <1>       ;
                                  <1>       ; OUTPUT ->
                                  <1>       ;     none
                                  <1>       ;
                                  <1>       ;     [u.pri] = priority of present process
                                  <1>       ;
                                  <1>       ;     cf = 1, if the request could not be fulfilled.
                                  <1>       ;
                                  <1>       ;     NOTE:
                                  <1>         ;          * Processes in 'run as regular' queue can run
                                  <1>       ;          if there is no process in 'run for event' queue
                                  <1>       ;          ('run for event' processes have higher priority)
                                  <1>       ;        * When [u.quant] time quantum of a process is
                                  <1>       ;          elapsed, it's high priority ('run for event')
                                  <1>       ;          status will be disabled, it can be run in sequence
                                  <1>       ;          of it's actual run queue.
                                  <1>       ;        * A 'run on background' process will always be
                                  <1>       ;          sequenced in 'run on background' (low priority)
                                  <1>       ;          queue, it can run only when other priority queues
                                  <1>       ;          are empty. (idle time processes, e.g. printing)
                                  <1>       ;
                                  <1>       ; Modified registers: eax, ebx, edx
                                  <1>       ;
                                  <1>
                                  <1> srunseq_0:
     0000F8EB 3A05[B3030300]      <1>         cmp     al, [u.uno]     ; same process ?
     0000F8F1 750C                <1>       jne   short srunseq_2 ; no
                                  <1>
     0000F8F3 8A25[A9030300]      <1>       mov   ah, [u.pri]  ; present/current priority
     0000F8F9 80FC02              <1>       cmp   ah, 2        ; 'run for event' priority level
     0000F8FC 7221                <1>       jb    short srunseq_6 ; no
                                  <1>
                                  <1> srunseq_1:
                                  <1>       ; there is nothing to do!
     0000F8FE C3                  <1>       retn
                                  <1>
                                  <1> srunseq_2:
                                  <1>       ;;this not necessary ! 23/12/2016
                                  <1>         ;;cmp      al, nproc    ; number of processes = 16
                                  <1>       ;;jnb  short srunseq_5     ; error ! invalid process number
                                  <1>
                                  <1>       ; dl = priority
     0000F8FF 80FA02              <1>       cmp   dl, 2        ; event queue
     0000F902 72FA                <1>       jb    short srunseq_1 ; requested process is not present
                                  <1>                           ; process and priority of requested
                                  <1>                           ; process is not high (event),
                                  <1>                           ; there is nothing to do!
                                  <1>
                                  <1>       ; requested process is not present process
                                  <1>       ; & priority of requested process is high
     0000F904 3A15[A9030300]      <1>       cmp   dl, [u.pri]  ; priority of present process
     0000F90A 7606                <1>       jna   short srunseq_3 ; is high, also
                                  <1>       ;
                                  <1>       ; present process will be swapped/switched out
     0000F90C FE05[CD650100]      <1>       inc   byte [p_change] ; 1
                                  <1>
                                  <1> srunseq_3:
                                  <1>       ; add process to 'runq_event' queue for new event
     0000F912 BB[52030300]        <1>       mov    ebx, runq_event ; high priority run queue
                                  <1>
                                  <1> srunseq_4:
                                  <1>       ; al = process number
                                  <1>       ; ebx = run queue
     0000F917 E881EDFFFF          <1>       call   putlu
     0000F91C C3                  <1>       retn
                                  <1>
                                  <1> srunseq_5:
     0000F91D F5                  <1>       cmc
     0000F91E C3                  <1>       retn
                                  <1>
                                  <1> srunseq_6:
                                  <1>       ; present priority of the process is not high
                                  <1>
     0000F91F 8815[A9030300]      <1>       mov    [u.pri], dl ; new priority
                                  <1>                       ; (will be used by 'tswap')
                                  <1>
     0000F925 80FA02              <1>       cmp    dl, 2        ; high priority ?
     0000F928 72F3                <1>       jb     short srunseq_5 ; no, there is nothing to do
                                  <1>                          ; in addition
                                  <1>
                                  <1>       ; process must be added to relevant run queue, here!
                                  <1>       ; (new priority is high/event priority and process
                                  <1>       ; will not be added to a run queue by 'tswap')
                                  <1>
     0000F92A BB[54030300]        <1>       mov    ebx, runq_normal ; 'run as regular' queue
                                  <1>
     0000F92F 20E4                <1>       and    ah, ah  ;  previous value of [u.pri]
     0000F931 75E4                <1>       jnz    short srunseq_4
                                  <1>
     0000F933 43                  <1>       inc    ebx
     0000F934 43                  <1>       inc    ebx
                                  <1>       ; ebx = runq_background ; 'run on backgroud' queue
                                  <1>
     0000F935 EBE0                <1>       jmp    short srunseq_4
                                  <1> clock:
                                  <1>       ; 23/05/2016
                                  <1>       ; 22/05/2016
                                  <1>       ; 20/05/2016
                                  <1>       ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
                                  <1>       ; 14/05/2015 - 14/10/2015 (Retro UNIX 386 v1)
                                  <1>       ; 07/12/2013 - 10/04/2014 (Retro UNIX 8086 v1)
                                  <1>
     0000F937 803D[A8030300]00    <1>       cmp    byte [u.quant], 0
     0000F93E 772C                <1>       ja     short clk_1
                                  <1>       ;
     0000F940 803D[B3030300]01    <1>       cmp     byte [u.uno], 1 ; /etc/init ? (for Retro UNIX 8086 & 386 v1)
                                  <1>                          ; MainProg (Kernel's Command Interpreter)
                                  <1>                          ; for TRDOS 386.
     0000F947 7623                <1>       jna    short clk_1 ; yes, do not swap out
                                  <1>       ;
     0000F949 803D[5B030300]FF    <1>       cmp     byte [sysflg], 0FFh ; user or system space ?
     0000F950 7520                <1>       jne    short clk_2      ; system space (sysflg <> 0FFh)
                                  <1>       ;
     0000F952 66833D[AA030300]00  <1>       cmp    word [u.intr], 0
     0000F95A 7616                <1>       jna    short clk_2
                                  <1>       ;
                                  <1>       ; 23/05/2016
     0000F95C 803D[CE650100]00    <1>       cmp    byte [multi_tasking], 0
     0000F963 760D                <1>       jna    short clk_2
                                  <1>       ;
     0000F965 FE05[CD650100]      <1>       inc    byte [p_change] ; it is time to change running process
     0000F96B C3                  <1>       retn
                                  <1> clk_1:
     0000F96C FE0D[A8030300]      <1>       dec    byte [u.quant]
                                  <1> clk_2:
     0000F972 C3                  <1>       retn   ; return to (hardware) timer interrupt routine
                                  <1>
                                  <1> ; 12/10/2017
                                  <1> ; 15/01/2017
                                  <1> ; 14/01/2017
                                  <1> ; 07/01/2017
                                  <1> ; 02/01/2017
                                  <1> ; 17/08/2016
                                  <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                                  <1> int34h:  ; #IOCTL# (I/O port access support for ring 3)
                                  <1> ; 23/05/2016
                                  <1>       ; 20/06/2016
                                  <1>       ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
                                  <1>       ;
                                  <1>       ; INPUT ->
                                  <1>       ;      AH = 0 -> read port  (physical IO port) -byte-
                                  <1>       ;      AH = 1 -> write port (physical IO port) -byte-
                                  <1>       ;            AL = data byte
                                  <1>       ;      AH = 2 -> read port  (physical IO port) -word-
                                  <1>       ;      AH = 3 -> write port (physical IO port) -word-
                                  <1>       ;            BX = data word
                                  <1>       ;      AH = 4 -> read port  (physical IO port) -dword-
                                  <1>       ;      AH = 5 -> write port (physical IO port) -dword-
                                  <1>       ;            EBX = data dword
                                  <1>       ;      ; 12/10/2017
                                  <1>       ;      AH = 6 -> read port (physical IO port) twice -byte-
                                  <1>       ;      AH = 7 -> write port (physical IO port) twice -byte-
                                  <1>       ;            BX = data word
                                  <1>       ;
                                  <1>       ;      DX = Port number (<= 0FFFFh)
                                  <1>       ;
                                  <1>       ; OUTPUT ->
                                  <1>       ;      AL = data byte (in al, dx)
                                  <1>       ;      AX = data word (in ax, dx)
                                  <1>       ;      EAX = data dword (in eax, dx)
                                  <1>       ;
                                  <1>       ;      (ECX = actual TRANSFER COUNT for string functions)
                                  <1>       ;
                                  <1>       ;
                                  <1>       ; Modified registers: EAX
                                  <1>       ;
                                  <1>
                                  <1>       ;cmp   ah, 5
                                  <1>       ;ja    short int34h_5 ; invalid function !
                                  <1>
                                  <1>       ; 12/10/2017
     0000F973 80FC07              <1>       cmp    ah, 7
     0000F976 7743                <1>       ja     short int34h_5 ; invalid function !
                                  <1>
                                  <1>       ;; 15/01/2017
                                  <1>       ; 14/01/2017
                                  <1>       ; 02/01/2017
                                  <1>       ;;mov  byte [ss:intflg], 34h     ; IOCTL interrupt
     0000F978 FB                  <1>       sti
                                  <1>
                                  <1>       ;sti   ; enable interrupts
     0000F979 80642408FE          <1>       and    byte [esp+8], 11111110b   ; clear carry bit of eflags register
                                  <1>
     0000F97E 80FC01              <1>       cmp    ah, 1
     0000F981 7205                <1>       jb     short int34h_0
     0000F983 7705                <1>       ja     short int34h_1
                                  <1>
     0000F985 EE                  <1>       out    dx, al
                                  <1>       ;iretd
     0000F986 EB01                <1>       jmp short int34h_iret
                                  <1>
                                  <1> int34h_0:
     0000F988 EC                  <1>       in     al, dx
                                  <1>       ;iretd
                                  <1> int34h_iret:
                                  <1>       ;cli   ; 07/01/2017
                                  <1>       ;; 15/01/2017
                                  <1>       ;;mov  byte [ss:intflg], 0 ; reset
     0000F989 CF                  <1>       iretd
                                  <1>
                                  <1> int34h_1:
     0000F98A F6C401              <1>       test   ah, 1
     0000F98D 7516                <1>       jnz    short int34h_3 ; out
                                  <1>
                                  <1>       ; in
     0000F98F 80FC02              <1>       cmp    ah, 2
     0000F992 7707                <1>       ja     short int34h_2
                                  <1>
     0000F994 6689D8              <1>       mov    ax, bx
     0000F997 66ED                <1>       in     ax, dx
                                  <1>       ;iretd
     0000F999 EBEE                <1>       jmp    short int34h_iret
                                  <1>
                                  <1> int34h_2:
     0000F99B 80FC04              <1>       cmp    ah, 4
     0000F99E 772C                <1>       ja     short int34h_7      ; 12/10/2017
                                  <1>       ; ah = 4
     0000F9A0 89D8                <1>       mov    eax, ebx
     0000F9A2 ED                  <1>       in     eax, dx
                                  <1>       ;iretd
     0000F9A3 EBE4                <1>       jmp    short int34h_iret
                                  <1>
                                  <1> int34h_3:
     0000F9A5 80FC03              <1>       cmp    ah, 3
     0000F9A8 7707                <1>       ja     short int34h_4
                                  <1>
     0000F9AA 6689D8              <1>       mov    ax, bx
     0000F9AD 66EF                <1>       out    dx, ax
                                  <1>       ;iretd
     0000F9AF EBD8                <1>       jmp    short int34h_iret
                                  <1>
                                  <1> int34h_4:
     0000F9B1 80FC05              <1>       cmp    ah, 5
     0000F9B4 770B                <1>       ja     short int34h_6      ; 12/10/2017
                                  <1>       ; ah = 5
     0000F9B6 89D8                <1>       mov    eax, ebx
     0000F9B8 EF                  <1>       out    dx, eax
                                  <1>       ;iretd
     0000F9B9 EBCE                <1>       jmp    short int34h_iret
                                  <1>
                                  <1> int34h_5:
     0000F9BB 804C240801          <1>       or     byte [esp+8], 1     ; set carry bit of eflags register
     0000F9C0 CF                  <1>       iretd
                                  <1>
                                  <1>       ; 12/10/2017
                                  <1> int34h_6:
     0000F9C1 6689D8              <1>       mov    ax, bx
     0000F9C4 EE                  <1>       out    dx, al
     0000F9C5 EB00                <1>       jmp    short $+2
     0000F9C7 86E0                <1>       xchg   ah, al
     0000F9C9 EE                  <1>       out    dx, al
                                  <1>       ;xchg  al, ah
                                  <1>       ;iretd
     0000F9CA EB06                <1>       jmp    short int34h_8
                                  <1> int34h_7:
     0000F9CC EC                  <1>       in     al, dx
     0000F9CD EB00                <1>       jmp    short $+2
     0000F9CF 88C4                <1>       mov    ah, al
     0000F9D1 EC                  <1>       in     al, dx
                                  <1> int34h_8:
     0000F9D2 86C4                <1>       xchg   al, ah
     0000F9D4 CF                  <1>       iretd
```

```
                             <1>
                             <1>
                             <1> INT4Ah:
                             <1>       ; 24/01/2016
                             <1>       ; this procedure will be called by 'RTC_INT' (in 'timer.s')
0000F9D5 C3                  <1>       retn
                             <1>
                             <1> ; u0.s
                             <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS0.INC
                             <1> ; Last Modification: 20/11/2015
                             <1>
                             <1> com2_int:
                             <1>       ; 07/11/2015
                             <1>       ; 24/10/2015
                             <1>       ; 23/10/2015
                             <1>       ; 14/03/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 28/07/2014 (Retro UNIX 8086 v1)
                             <1>       ; < serial port 2 interrupt handler >
                             <1>       ;
0000F9D6 890424              <1>       mov    [esp], eax ; overwrite call return address
                             <1>       ;push  eax
0000F9D9 66B80900            <1>       mov    ax, 9
0000F9DD EB07                <1>       jmp    short comm_int
                             <1> com1_int:
                             <1>       ; 07/11/2015
                             <1>       ; 24/10/2015
0000F9DF 890424              <1>       mov    [esp], eax ; overwrite call return address
                             <1>       ; 23/10/2015
                             <1>       ;push  eax
0000F9E2 66B80800            <1>       mov    ax, 8
                             <1> comm_int:
                             <1>       ; 20/11/2015
                             <1>       ; 18/11/2015
                             <1>       ; 17/11/2015
                             <1>       ; 16/11/2015
                             <1>       ; 09/11/2015
                             <1>       ; 08/11/2015
                             <1>       ; 07/11/2015
                             <1>       ; 06/11/2015 (serial4.asm, 'serial')
                             <1>       ; 01/11/2015
                             <1>       ; 26/10/2015
                             <1>       ; 23/10/2015
0000F9E6 53                  <1>       push   ebx
0000F9E7 56                  <1>       push   esi
0000F9E8 57                  <1>       push   edi
0000F9E9 1E                  <1>       push   ds
0000F9EA 06                  <1>       push   es
                             <1>       ; 18/11/2015
0000F9EB 0F20DB              <1>       mov    ebx, cr3
0000F9EE 53                  <1>       push   ebx ; ****
                             <1>       ;
0000F9EF 51                  <1>       push   ecx ; ***
0000F9F0 52                  <1>       push   edx ; **
                             <1>       ;
0000F9F1 BB10000000          <1>       mov    ebx, KDATA
0000F9F6 8EDB                <1>       mov    ds, bx
0000F9F8 8EC3                <1>       mov    es, bx
                             <1>       ;
0000F9FA 8B0D[38580100]      <1>       mov    ecx, [k_page_dir]
0000FA00 0F22D9              <1>       mov    cr3, ecx
                             <1>       ; 20/11/2015
                             <1>       ; Interrupt identification register
0000FA03 66BAFA02            <1>       mov    dx, 2FAh ; COM2
                             <1>       ;
0000FA07 3C08                <1>       cmp    al, 8
0000FA09 7702                <1>       ja     short com_i0
                             <1>       ;
                             <1>       ; 20/11/2015
                             <1>       ; 17/11/2015
                             <1>       ; 16/11/2015
                             <1>       ; 15/11/2015
                             <1>       ; 24/10/2015
                             <1>       ; 14/03/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>       ; 28/07/2014 (Retro UNIX 8086 v1)
                             <1>       ; < serial port 1 interrupt handler >
                             <1>       ;
0000FA0B FEC6                <1>       inc    dh ; 3FAh ; COM1 Interrupt id. register
                             <1> com_i0:
                             <1>       ;push  eax ; *
                             <1>       ; 07/11/2015
0000FA0D A2[A2580100]        <1>       mov    byte [ccomport], al
                             <1>       ; 09/11/2015
0000FA12 0FB7D8              <1>       movzx  ebx, ax ; 8 or 9
                             <1>       ; 17/11/2015
                             <1>       ; reset request for response status
0000FA15 88A3[98580100]      <1>       mov    [ebx+req_resp-8], ah ; 0
                             <1>       ;
                             <1>       ; 20/11/2015
0000FA1B EC                  <1>       in     al, dx       ; read interrupt id. register
0000FA1C EB00                <1>       JMP    $+2          ; I/O DELAY
0000FA1E 2404                <1>       and    al, 4        ; received data available?
0000FA20 7470                <1>       jz     short com_eoi; (transmit. holding reg. empty)
                             <1>       ;
                             <1>       ; 20/11/2015
0000FA22 80EA02              <1>       sub    dl, 3FAh-3F8h; data register (3F8h, 2F8h)
0000FA25 EC                  <1>       in     al, dx       ; read character
                             <1>       ;JMP   $+2          ; I/O DELAY
                             <1>       ; 08/11/2015
                             <1>       ; 07/11/2015
0000FA26 89DE                <1>       mov    esi, ebx
0000FA28 89DF                <1>       mov    edi, ebx
0000FA2A 81C6[9C580100]      <1>       add    esi, rchar - 8 ; points to last received char
0000FA30 81C7[9E580100]      <1>       add    edi, schar - 8 ; points to last sent char
```

```
0000FA36 8806                <1>       mov    [esi], al ; received char (current char)
                             <1>       ; query
0000FA38 20C0                <1>       and    al, al
0000FA3A 7527                <1>       jnz    short com_i2
                             <1>       ; response
                             <1>       ; 17/11/2015
                             <1>       ; set request for response status
0000FA3C FE83[98580100]      <1>         inc     byte [ebx+req_resp-8] ; 1
                             <1>       ;
0000FA42 6683C205            <1>       add    dx, 3FDh-3F8h; (3FDh, 2FDh)
0000FA46 EC                  <1>       in     al, dx       ; read line status register
0000FA47 EB00                <1>       JMP    $+2          ; I/O DELAY
0000FA49 2420                <1>       and    al, 20h             ; transmitter holding reg. empty?
0000FA4B 7445                <1>       jz     short com_eoi       ; no
0000FA4D B0FF                <1>       mov    al, 0FFh     ; response
0000FA4F 6683EA05            <1>       sub    dx, 3FDh-3F8h       ; data port (3F8h, 2F8h)
0000FA53 EE                  <1>       out    dx, al       ; send on serial port
                             <1>       ; 17/11/2015
0000FA54 803F00              <1>       cmp    byte [edi], 0   ; query ? (schar)
0000FA57 7502                <1>       jne    short com_i1    ; no
0000FA59 8807                <1>       mov    [edi], al    ; 0FFh (responded)
                             <1> com_i1:
                             <1>       ; 17/11/2015
                             <1>       ; reset request for response status (again)
0000FA5B FE8B[98580100]      <1>         dec     byte [ebx+req_resp-8] ; 0
0000FA61 EB2F                <1>       jmp    short com_eoi
                             <1> com_i2:
                             <1>       ; 08/11/2015
0000FA63 3CFF                <1>       cmp    al, 0FFh     ; (response ?)
0000FA65 7417                <1>       je     short com_i3 ; (check for response signal)
                             <1>       ; 07/11/2015
0000FA67 3C04                <1>       cmp    al, 04h      ; EOT
0000FA69 751C                <1>       jne    short com_i4
                             <1>       ; EOT = 04h (End of Transmit) - 'CTRL + D'
                             <1>       ;(an EOT char is supposed as a ctrl+brk from the terminal)
                             <1>       ; 08/11/2015
                             <1>              ; ptty -> tty 0 to 7 (pseudo screens)
0000FA6B 861D[66580100]      <1>       xchg   bl, [ptty]  ; tty number (8 or 9)
0000FA71 E86069FFFF          <1>       call   ctrlbrk
0000FA76 861D[66580100]      <1>       xchg   [ptty], bl ; (restore ptty value and BL value)
                             <1>       ;mov   al, 04h ; EOT
                             <1>       ; 08/11/2015
0000FA7C EB09                <1>       jmp    short com_i4
                             <1> com_i3:
                             <1>       ; 08/11/2015
                             <1>       ; If 0FFh has been received just after a query
                             <1>       ; (schar, ZERO), it is a response signal.
                             <1>       ; 17/11/2015
0000FA7E 803F00              <1>         cmp     byte [edi], 0 ; query ? (schar)
0000FA81 7704                <1>       ja     short com_i4 ; no
                             <1>       ; reset query status (schar)
0000FA83 8807                <1>       mov    [edi], al ; 0FFh
0000FA85 FEC0                <1>       inc    al ; 0
                             <1> com_i4:
                             <1>       ; 27/07/2014
                             <1>       ; 09/07/2014
0000FA87 D0E3                <1>       shl    bl, 1
0000FA89 81C3[68580100]      <1>       add    ebx, ttychr
                             <1>       ; 23/07/2014 (always overwrite)
                             <1>       ;;cmp  word [ebx], 0
                             <1>       ;;ja   short com_eoi
                             <1>       ;
0000FA8F 668903              <1>       mov    [ebx], ax   ; Save ascii code
                             <1>                          ; scan code = 0
                             <1> com_eoi:
                             <1>       ;mov   al, 20h
                             <1>       ;out   20h, al         ; end of interrupt
                             <1>       ;
                             <1>       ; 07/11/2015
                             <1>              ;pop   eax ; *
0000FA92 A0[A2580100]        <1>       mov    al, byte [ccomport] ; current COM port
                             <1>        ; al = tty number (8 or 9)
0000FA97 E85E010000          <1>         call wakeup
                             <1> com_iret:
                             <1>       ; 23/10/2015
0000FA9C 5A                  <1>       pop    edx ; **
0000FA9D 59                  <1>       pop    ecx ; ***
                             <1>       ; 18/11/2015
                             <1>       ;pop   eax ; ****
                             <1>       ;mov   cr3, eax
                             <1>       ;jmp   iiret
0000FA9E E93D10FFFF          <1>       jmp    iiretp
                             <1>
                             <1> ;iiretp: ; 01/09/2015
                             <1> ;     ; 28/08/2015
                             <1> ;     pop    eax ; (*) page directory
                             <1> ;     mov    cr3, eax
                             <1> ;iiret:
                             <1> ;     ; 22/08/2014
                             <1> ;     mov    al, 20h ; END OF INTERRUPT COMMAND TO 8259
                             <1> ;     out    20h, al      ; 8259 PORT
                             <1> ;     ;
                             <1> ;     pop    es
                             <1> ;     pop    ds
                             <1> ;     pop    edi
                             <1> ;     pop    esi
                             <1> ;     pop    ebx ; 29/08/2014
                             <1> ;     pop    eax
                             <1> ;     iretd
                             <1>
                             <1> sp_init:
                             <1>       ; 07/11/2015
                             <1>       ; 29/10/2015
```

```
                              <1>       ; 26/10/2015
                              <1>       ; 23/10/2015
                              <1>       ; 29/06/2015
                              <1>       ; 14/03/2015 (Retro UNIX 386 v1 - 115200 baud)
                              <1>       ; 28/07/2014 (Retro UNIX 8086 v1 - 9600 baud)
                              <1>       ; Initialization of Serial Port Communication Parameters
                              <1>       ; (COM1 base port address = 3F8h, COM1 Interrupt = IRQ 4)
                              <1>       ; (COM2 base port address = 2F8h, COM1 Interrupt = IRQ 3)
                              <1>       ;
                              <1>       ; ((Modified registers: EAX, ECX, EDX, EBX))
                              <1>       ;
                              <1>       ; INPUT:  (29/06/2015)
                              <1>       ;     AL = 0 for COM1
                              <1>       ;          1 for COM2
                              <1>       ;     AH = Communication parameters
                              <1>       ;
                              <1>       ;  (*) Communication parameters (except BAUD RATE):
                              <1>       ;     Bit    4      3      2      1      0
                              <1>       ;            -PARITY--   STOP BIT  -WORD LENGTH-
                              <1>       ;  this one -->     00 = none    0 = 1 bit  11 = 8 bits
                              <1>       ;            01 = odd     1 = 2 bits   10 = 7 bits
                              <1>       ;            11 = even
                              <1>       ;  Baud rate setting bits: (29/06/2015)
                              <1>       ;            Retro UNIX 386 v1 feature only !
                              <1>       ;     Bit    7    6    5  | Baud rate
                              <1>       ;            ------------------------
                              <1>       ;     value  0    0    0  | Default (Divisor = 1)
                              <1>       ;            0    0    1  | 9600 (12)
                              <1>       ;            0    1    0  | 19200 (6)
                              <1>       ;            0    1   1  | 38400 (3)
                              <1>       ;            1    0   0  | 14400 (8)
                              <1>       ;            1    0   1  | 28800 (4)
                              <1>       ;            1    1    0  | 57600 (2)
                              <1>       ;            1    1    1  | 115200 (1)
                              <1>
                              <1>       ; References:
                              <1>       ; (1) IBM PC-XT Model 286 BIOS Source Code
                              <1>       ;     RS232.ASM --- 10/06/1985 COMMUNICATIONS BIOS (RS232)
                              <1>       ; (2) Award BIOS 1999 - ATORGS.ASM
                              <1>       ; (3) http://wiki.osdev.org/Serial_Ports
                              <1>       ;
                              <1>       ; Set communication parameters for COM1 (= 03h)
                              <1>       ;
0000FAA3 BB[9E580100]         <1>       mov    ebx, com1p          ; COM1 parameters
0000FAA8 66BAF803             <1>       mov    dx, 3F8h            ; COM1
                              <1>        ; 29/10/2015
0000FAAC 66B90103             <1>       mov    cx, 301h  ; divisor = 1 (115200 baud)
0000FAB0 E86F000000           <1>       call   sp_i3  ; call A4
0000FAB5 A880                 <1>       test   al, 80h
0000FAB7 7410                 <1>       jz     short sp_i0 ; OK..
                              <1>              ; Error !
                              <1>       ;mov   dx, 3F8h
0000FAB9 80EA05               <1>       sub    dl, 5 ; 3FDh -> 3F8h
0000FABC 66B90E03             <1>       mov    cx, 30Eh  ; divisor = 12 (9600 baud)
0000FAC0 E85F000000           <1>       call   sp_i3  ; call A4
0000FAC5 A880                 <1>       test   al, 80h
0000FAC7 7508                 <1>       jnz    short sp_i1
                              <1> sp_i0:
                              <1>         ; (Note: Serial port interrupts will be disabled here...)
                              <1>         ; (INT 14h initialization code disables interrupts.)
                              <1>       ;
0000FAC9 C603E3               <1>       mov    byte [ebx], 0E3h ; 11100011b
0000FACC E8DC000000           <1>       call   sp_i5 ; 29/06/2015
                              <1> sp_i1:
0000FAD1 43                   <1>       inc    ebx
0000FAD2 66BAF802             <1>       mov    dx, 2F8h            ; COM2
                              <1>        ; 29/10/2015
0000FAD6 66B90103             <1>       mov    cx, 301h  ; divisor = 1 (115200 baud)
0000FADA E845000000           <1>       call   sp_i3  ; call A4
0000FADF A880                 <1>       test   al, 80h
0000FAE1 7410                 <1>       jz     short sp_i2 ; OK..
                              <1>              ; Error !
                              <1>       ;mov   dx, 2F8h
0000FAE3 80EA05               <1>       sub    dl, 5 ; 2FDh -> 2F8h
0000FAE6 66B90E03             <1>       mov    cx, 30Eh  ; divisor = 12 (9600 baud)
0000FAEA E835000000           <1>       call   sp_i3  ; call A4
0000FAEF A880                 <1>       test   al, 80h
0000FAF1 7530                 <1>       jnz    short sp_i7
                              <1> sp_i2:
0000FAF3 C603E3               <1>       mov    byte [ebx], 0E3h ; 11100011b
                              <1> sp_i6:
                              <1>       ;; COM2 - enabling IRQ 3
                              <1>       ; 07/11/2015
                              <1>       ; 26/10/2015
0000FAF6 9C                   <1>       pushf
0000FAF7 FA                   <1>       cli
                              <1>       ;
0000FAF8 66BAFC02             <1>       mov    dx, 2FCh            ; modem control register
0000FAFC EC                   <1>       in     al, dx                    ; read register
0000FAFD EB00                 <1>       JMP    $+2                 ; I/O DELAY
0000FAFF 0C08                 <1>       or     al, 8               ; enable bit 3 (OUT2)
0000FB01 EE                   <1>       out    dx, al              ; write back to register
0000FB02 EB00                 <1>       JMP    $+2                 ; I/O DELAY
0000FB04 66BAF902             <1>       mov    dx, 2F9h            ; interrupt enable register
0000FB08 EC                   <1>       in     al, dx              ; read register
0000FB09 EB00                 <1>       JMP    $+2                 ; I/O DELAY
                              <1>       ;or    al, 1               ; receiver data interrupt enable and
0000FB0B 0C03                 <1>       or     al, 3               ; transmitter empty interrupt enable
0000FB0D EE                   <1>       out    dx, al                    ; write back to register
0000FB0E EB00                 <1>       JMP    $+2                 ; I/O DELAY
0000FB10 E421                 <1>       in     al, 21h             ; read interrupt mask register
0000FB12 EB00                 <1>       JMP    $+2                 ; I/O DELAY
0000FB14 24F7                 <1>       and    al, 0F7h            ; enable IRQ 3 (COM2)
```

```
0000FB16 E621                <1>       out    21h, al             ; write back to register
                             <1>       ;
                             <1>       ; 23/10/2015
0000FB18 B8[D6F90000]        <1>       mov    eax, com2_int
0000FB1D A3[F5FB0000]        <1>       mov    [com2_irq3], eax
                             <1>       ; 26/10/2015
0000FB22 9D                  <1>       popf
                             <1> sp_i7:
0000FB23 C3                  <1>       retn
                             <1>
                             <1> sp_i3:
                             <1> ;A4:         ;----- INITIALIZE THE COMMUNICATIONS PORT
                             <1>       ; 28/10/2015
0000FB24 FEC2                <1>       inc    dl     ; 3F9h (2F9h); 3F9h, COM1 Interrupt enable register
0000FB26 B000                <1>       mov    al, 0
0000FB28 EE                  <1>       out    dx, al              ; disable serial port interrupt
0000FB29 EB00                <1>       JMP    $+2                 ; I/O DELAY
0000FB2B 80C202              <1>       add    dl, 2  ; 3FBh (2FBh); COM1 Line control register (3FBh)
0000FB2E B080                <1>       mov    al, 80h
0000FB30 EE                  <1>       out    dx, al              ; SET DLAB=1 ; divisor latch access bit
                             <1>       ;----- SET BAUD RATE DIVISOR
                             <1>       ; 26/10/2015
0000FB31 80EA03              <1>       sub    dl, 3   ; 3F8h (2F8h)     ; register for least significant byte
                             <1>                                  ; of the divisor value
0000FB34 88C8                <1>       mov    al, cl ; 1
0000FB36 EE                  <1>       out    dx, al              ; 1 = 115200 baud (Retro UNIX 386 v1)
                             <1>                                  ; 2 = 57600 baud
                             <1>                                  ; 3 = 38400 baud
                             <1>                                  ; 6 = 19200 baud
                             <1>                                  ; 12 = 9600 baud (Retro UNIX 8086 v1)
0000FB37 EB00                <1>       JMP    $+2                 ; I/O DELAY
0000FB39 28C0                <1>       sub    al, al
0000FB3B FEC2                <1>       inc    dl      ; 3F9h (2F9h)     ; register for most significant byte
                             <1>                                  ; of the divisor value
0000FB3D EE                  <1>       out    dx, al ; 0
0000FB3E EB00                <1>       JMP    $+2                 ; I/O DELAY
                             <1>       ;
0000FB40 88E8                <1>       mov    al, ch ; 3          ; 8 data bits, 1 stop bit, no parity
                             <1>       ;and   al, 1Fh ; Bits 0,1,2,3,4
0000FB42 80C202              <1>       add    dl, 2  ; 3FBh (2FBh); Line control register
0000FB45 EE                  <1>       out    dx, al
0000FB46 EB00                <1>       JMP    $+2                 ; I/O DELAY
                             <1>       ; 29/10/2015
0000FB48 FECA                <1>       dec    dl     ; 3FAh (2FAh); FIFO Control register (16550/16750)
0000FB4A 30C0                <1>       xor    al, al              ; 0
0000FB4C EE                  <1>       out    dx, al              ; Disable FIFOs (reset to 8250 mode)
0000FB4D EB00                <1>       JMP    $+2
                             <1> sp_i4:
                             <1> ;A18: ;----- COMM PORT STATUS ROUTINE
                             <1>       ; 29/06/2015 (line status after modem status)
0000FB4F 80C204              <1>       add    dl, 4  ; 3FEh (2FEh); Modem status register
                             <1> sp_i4s:
0000FB52 EC                  <1>       in     al, dx              ; GET MODEM CONTROL STATUS
0000FB53 EB00                <1>       JMP    $+2                 ; I/O DELAY
0000FB55 88C4                <1>       mov    ah, al              ; PUT IN (AH) FOR RETURN
0000FB57 FECA                <1>       dec    dl     ; 3FDh (2FDh); POINT TO LINE STATUS REGISTER
                             <1>                                  ; dx = 3FDh for COM1, 2FDh for COM2
0000FB59 EC                  <1>       in     al, dx              ; GET LINE CONTROL STATUS
                             <1>       ; AL = Line status, AH = Modem status
0000FB5A C3                  <1>       retn
                             <1>
                             <1> sp_status:
                             <1>       ; 29/06/2015
                             <1>       ; 27/06/2015 (Retro UNIX 386 v1)
                             <1>       ; Get serial port status
0000FB5B 66BAFE03            <1>       mov    dx, 3FEh            ; Modem status register (COM1)
0000FB5F 28C6                <1>       sub    dh, al              ; dh = 2 for COM2 (al = 1)
                             <1>                                  ; dx = 2FEh for COM2
0000FB61 EBEF                <1>       jmp    short sp_i4s
                             <1>
                             <1> sp_setp: ; Set serial port communication parameters
                             <1>       ; 07/11/2015
                             <1>       ; 29/10/2015
                             <1>       ; 29/06/2015
                             <1>       ; Retro UNIX 386 v1 feature only !
                             <1>       ;
                             <1>       ; INPUT:
                             <1>       ;      AL = 0 for COM1
                             <1>       ;           1 for COM2
                             <1>       ;      AH = Communication parameters (*)
                             <1>       ; OUTPUT:
                             <1>       ;      CL = Line status
                             <1>       ;      CH = Modem status
                             <1>       ;   If cf = 1 -> Error code in [u.error]
                             <1>       ;             'invalid parameter !'
                             <1>       ;                    or
                             <1>       ;             'device not ready !' error
                             <1>       ;
                             <1>       ;  (*) Communication parameters (except BAUD RATE):
                             <1>       ;      Bit    4      3      2      1      0
                             <1>       ;             -PARITY--    STOP BIT  -WORD LENGTH-
                             <1>       ;  this one -->     00 = none    0 = 1 bit  11 = 8 bits
                             <1>       ;             01 = odd      1 = 2 bits   10 = 7 bits
                             <1>       ;             11 = even
                             <1>       ;  Baud rate setting bits: (29/06/2015)
                             <1>       ;             Retro UNIX 386 v1 feature only !
                             <1>       ;      Bit    7    6    5  | Baud rate
                             <1>       ;             ------------------------
                             <1>       ;      value  0    0    0  | Default (Divisor = 1)
                             <1>       ;             0    0    1  | 9600 (12)
                             <1>       ;             0    1    0  | 19200 (6)
                             <1>       ;             0    1   1  | 38400 (3)
                             <1>       ;             1    0   0  | 14400 (8)
                             <1>       ;            1    0   1  | 28800 (4)
                             <1>       ;            1    1    0  | 57600 (2)
                             <1>       ;            1    1    1  | 115200 (1)
                             <1>       ;
                             <1>       ; (COM1 base port address = 3F8h, COM1 Interrupt = IRQ 4)
                             <1>       ; (COM2 base port address = 2F8h, COM1 Interrupt = IRQ 3)
                             <1>       ;
                             <1>       ; ((Modified registers: EAX, ECX, EDX, EBX))
                             <1>       ;
0000FB63 66BAF803            <1>       mov    dx, 3F8h
0000FB67 BB[9E580100]        <1>       mov    ebx, com1p ; COM1 control byte offset
0000FB6C 3C01                <1>       cmp    al, 1
0000FB6E 776B                <1>       ja     short sp_invp_err
0000FB70 7203                <1>       jb     short sp_setp1 ;  COM1 (AL = 0)
0000FB72 FECE                <1>       dec    dh ; 2F8h
0000FB74 43                  <1>       inc    ebx ; COM2 control byte offset
                             <1> sp_setp1:
                             <1>       ; 29/10/2015
0000FB75 8823                <1>       mov    [ebx], ah
0000FB77 0FB6CC              <1>       movzx  ecx, ah
0000FB7A C0E905              <1>       shr    cl, 5 ; -> baud rate index
0000FB7D 80E41F              <1>       and    ah, 1Fh ; communication parameters except baud rate
0000FB80 8A81[EAFB0000]      <1>       mov    al, [ecx+b_div_tbl]
0000FB86 6689C1              <1>       mov    cx, ax
0000FB89 E896FFFFFF          <1>       call   sp_i3
0000FB8E 6689C1              <1>       mov    cx, ax ; CL = Line status, CH = Modem status
0000FB91 A880                <1>       test   al, 80h
0000FB93 740F                <1>       jz     short sp_setp2
0000FB95 C603E3              <1>         mov     byte [ebx], 0E3h ; Reset to initial value (11100011b)
                             <1> stp_dnr_err:
0000FB98 C705[C8030300]0F00- <1>       mov    dword [u.error], ERR_DEV_NOT_RDY ; 'device not ready !'
0000FBA0 0000                <1>
                             <1>       ; CL = Line status, CH = Modem status
0000FBA2 F9                  <1>       stc
0000FBA3 C3                  <1>       retn
                             <1> sp_setp2:
0000FBA4 80FE02              <1>       cmp    dh, 2 ; COM2 (2F?h)
0000FBA7 0F8649FFFFFF        <1>         jna     sp_i6
                             <1>                    ; COM1 (3F?h)
                             <1> sp_i5:
                             <1>       ; 07/11/2015
                             <1>       ; 26/10/2015
                             <1>       ; 29/06/2015
                             <1>       ;
                             <1>       ;; COM1 - enabling IRQ 4
0000FBAD 9C                  <1>       pushf
0000FBAE FA                  <1>       cli
0000FBAF 66BAFC03            <1>       mov    dx, 3FCh           ; modem control register
0000FBB3 EC                  <1>       in     al, dx                    ; read register
0000FBB4 EB00                <1>       JMP    $+2                ; I/O DELAY
0000FBB6 0C08                <1>       or     al, 8              ; enable bit 3 (OUT2)
0000FBB8 EE                  <1>       out    dx, al             ; write back to register
0000FBB9 EB00                <1>       JMP    $+2                ; I/O DELAY
0000FBBB 66BAF903            <1>       mov    dx, 3F9h           ; interrupt enable register
0000FBBF EC                  <1>       in     al, dx             ; read register
0000FBC0 EB00                <1>       JMP    $+2                ; I/O DELAY
                             <1>       ;or    al, 1              ; receiver data interrupt enable and
0000FBC2 0C03                <1>       or     al, 3              ; transmitter empty interrupt enable
0000FBC4 EE                  <1>       out    dx, al                    ; write back to register
0000FBC5 EB00                <1>       JMP    $+2                ; I/O DELAY
0000FBC7 E421                <1>       in     al, 21h            ; read interrupt mask register
0000FBC9 EB00                <1>       JMP    $+2                ; I/O DELAY
0000FBCB 24EF                <1>       and    al, 0EFh           ; enable IRQ 4 (COM1)
0000FBCD E621                <1>       out    21h, al            ; write back to register
                             <1>       ;
                             <1>       ; 23/10/2015
0000FBCF B8[DFF90000]        <1>       mov    eax, com1_int
0000FBD4 A3[F1FB0000]        <1>       mov    [com1_irq4], eax
                             <1>       ; 26/10/2015
0000FBD9 9D                  <1>       popf
0000FBDA C3                  <1>       retn
                             <1>
                             <1> sp_invp_err:
0000FBDB C705[C8030300]1700- <1>       mov    dword [u.error], ERR_INV_PARAMETER ; 'invalid parameter !'
0000FBE3 0000                <1>
0000FBE5 31C9                <1>       xor    ecx, ecx
0000FBE7 49                  <1>       dec    ecx ; 0FFFFh
0000FBE8 F9                  <1>       stc
0000FBE9 C3                  <1>       retn
                             <1>
                             <1> ; 29/10/2015
                             <1> b_div_tbl: ; Baud rate divisor table (115200/divisor)
0000FBEA 010C0603080401      <1>       db 1, 12, 6, 3, 8, 4, 1
                             <1>
                             <1>
                             <1> ; 23/10/2015
                             <1> com1_irq4:
0000FBF1 [F9FB0000]          <1>       dd dummy_retn
                             <1> com2_irq3:
0000FBF5 [F9FB0000]          <1>       dd dummy_retn
                             <1>
                             <1> dummy_retn:
0000FBF9 C3                  <1>       retn
                             <1>
                             <1> wakeup:
                             <1>       ; 24/01/2016
0000FBFA C3                  <1>       retn
                             <1>
                             <1> set_working_path_x:
                             <1>              ; 17/10/2016 (TRDOS 386 - FFF & FNF)
0000FBFB 66B80100            <1>              mov    ax, 1
                             <1>                     ; File name is needed/forced (AL=1)
                             <1>                     ; Change directory as temporary (AH=0)
                             <1>
                             <1> set_working_path_xx: ; 30/12/2017 (syschdir)
                             <1>             ; This is needed for preventing wrong Find Next File
                             <1>             ; system call after sysopen, syscreate, sysmkdir etc.
                             <1>             ; Find Next File must immediate follow Find First File)
                             <1>
0000FBFF 8825[F0650100]      <1>             mov    [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
                             <1>
                             <1> set_working_path:
                             <1>             ; 16/10/2016
                             <1>             ; 12/10/2016
                             <1>             ; 10/10/2016
                             <1>             ; 05/10/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>             ;
                             <1>             ; TRDOS v1.0 (DIR.ASM, "proc_set_working_path")
                             <1>                 ; 27/01/2011 - 08/02/2011
                             <1>             ; Set/Changes current drive, directory and file
                             <1>             ; depending on command tail
                             <1>             ; (procedure is derivated from CMD_INTR.ASM
                             <1>             ; file or dir locating code of internal commands)
                             <1>             ; (This procedure is prepared for INT 21H file/dir
                             <1>             ; functions and also to get compact code for
                             <1>             ; internal mainprog -command interpreter- commands)
                             <1>             ;
                             <1>             ; INPUT: DS:SI -> Command tail (ASCIIZ string)
                             <1>             ; AL = 0 -> any, AL > 0 -> file name is forced
                             <1>             ; AH = CD -> Change directory permanently
                             <1>             ; AH <> CD -> Change directory as temporary
                             <1>             ;
                             <1>             ; OUTPUT: ES=DS, FindFile structure has been set
                             <1>             ;         RUN_CDRV points previous current drive
                             <1>             ;         DS:SI = FindFile structure address
                             <1>             ;         (DS=CS)
                             <1>             ;         AX, BX, CX, DX, DI will be changed
                             <1>             ;    cf = 1 -> Error code in AX (AL)
                             <1>             ;         stc & AX = 0 -> Bad command or path name
                             <1>             ; ----------------------------------------------
                             <1>             ;
                             <1>             ; TRDOS 386 (05/10/2016)
                             <1>             ; INPUT:
                             <1>             ;      ESI = File/Directory Path (ASCIIZ string)
                             <1>             ;             address in user's memory space
                             <1>             ;       AL = 0 -> any
                             <1>             ;       AL > 0 -> file name is forced
                             <1>             ;       AH = CD -> change directory as permanent
                             <1>             ;       AH <> CD -> change directory as temporary
                             <1>             ;
                             <1>             ; OUTPUT:
                             <1>             ;      FindFile structure has been set
                             <1>             ;       RUN_CDRV points previous current drive
                             <1>             ;       ESI = FindFile_Name address ; 12/10/2016
                             <1>             ;
                             <1>             ;       cf = 1 -> Error code in EAX (AL)
                             <1>             ;       stc & EAX = 0 -> Bad command or path name
                             <1>             ;
                             <1>             ; Modified registers: EAX, EBX, ECX, EDX, ESI, EDI
                             <1>
0000FC05 66A3[F4650100]      <1>             mov    [SWP_Mode], ax
0000FC0B A0[FE580100]        <1>             mov    al, [Current_Drv]
0000FC10 30E4                <1>             xor    ah, ah
0000FC12 66A3[F6650100]      <1>             mov    [SWP_DRV], ax
                             <1>
                             <1>             ; TRDOS 386 ring 3 (user's page directory)
                             <1>             ; to ring 0 (kernel's page directory)
                             <1>             ; transfer modifications (05/10/2016).
                             <1>
0000FC18 55                  <1>             push   ebp
0000FC19 89E5                <1>             mov    ebp, esp
                             <1>
0000FC1B B980000000          <1>             mov    ecx, 128 ; maximum path length = 128 bytes
0000FC20 29CC                <1>             sub    esp, ecx ; reserve 128 bytes (buffer) on stack
0000FC22 89E7                <1>             mov    edi, esp ; destination address (kernel space)
                             <1>             ; esi = source address (virtual, in user's memory space)
0000FC24 E89AEBFFFF          <1>             call   transfer_from_user_buffer
0000FC29 720A                <1>             jc     short loc_swp_xor_retn
                             <1>
0000FC2B 89E6                <1>             mov    esi, esp ; temporary buffer (the path) on stack
                             <1> loc_swp_fchar:
0000FC2D 8A06                <1>             mov    al, [esi]
0000FC2F 3C20                <1>             cmp    al, 20h
0000FC31 7711                <1>             ja     short loc_swp_parse_path_name
0000FC33 740C                <1>             je     short loc_swp_fchar_next
                             <1>
                             <1> loc_swp_xor_retn:
0000FC35 31C0                <1>             xor    eax, eax
0000FC37 F9                  <1>             stc
                             <1> loc_swp_retn:
0000FC38 89EC                <1>             mov    esp, ebp
0000FC3A 5D                  <1>             pop    ebp
                             <1>
                             <1>             ;mov   esi, FindFile_Drv
0000FC3B BE[E4620100]        <1>             mov    esi, FindFile_Name ; 12/10/2016
0000FC40 C3                  <1>             retn
                             <1>
                             <1> loc_swp_fchar_next:
0000FC41 46                  <1>             inc    esi
0000FC42 EBE9                <1>             jmp    short loc_swp_fchar
                             <1>
                             <1> loc_swp_parse_path_name:
0000FC44 BF[A2620100]        <1>             mov    edi, FindFile_Drv
0000FC49 E8E6A7FFFF          <1>             call   parse_path_name
0000FC4E 72E8                <1>             jc     short loc_swp_retn
                             <1>
```

```
                                 <1> loc_swp_checkfile_name:
0000FC50 803D[F4650100]00        <1>             cmp    byte [SWP_Mode], 0
0000FC57 761E                    <1>             jna    short loc_swp_drv
                                 <1> 
                                 <1>             ; 10/10/2016 (valid file name checking)
0000FC59 BE[E4620100]            <1>             mov    esi, FindFile_Name
0000FC5E 803E20                  <1>             cmp    byte [esi], 20h
0000FC61 76D2                    <1>             jna    short loc_swp_xor_retn
                                 <1> 
                                 <1>             ; 16/10/2016
0000FC63 C605[F3650100]00        <1>             mov    byte [SWP_inv_fname], 0 ; reset
                                 <1>             ; esi = file name address (ASCIIZ)
0000FC6A E8B289FFFF              <1>             call   check_filename
0000FC6F 7306                    <1>             jnc    short loc_swp_drv
                                 <1> 
0000FC71 FE05[F3650100]          <1>             inc    byte [SWP_inv_fname] ; set
                                 <1> loc_swp_drv:
0000FC77 8A35[FE580100]          <1>             mov    dh, [Current_Drv]
                                 <1>                ;mov       [RUN_CDRV], dh
                                 <1> 
0000FC7D 8A15[A2620100]          <1>             mov    dl, [FindFile_Drv]
                                 <1>                ;cmp       dl, dh
0000FC83 3A15[FE580100]          <1>             cmp    dl, [Current_Drv]
0000FC89 740D                    <1>             je     short loc_swp_change_directory
                                 <1> 
0000FC8B FE05[F7650100]          <1>             inc    byte [SWP_DRV_chg]
0000FC91 E82A72FFFF              <1>             call   change_current_drive
0000FC96 72A0                    <1>             jc     short loc_swp_retn ; eax = error code
                                 <1>             ; eax = 0
                                 <1> 
                                 <1> loc_swp_change_directory:
0000FC98 803D[A3620100]21        <1>             cmp    byte [FindFile_Directory], 21h
0000FC9F F5                      <1>             cmc
0000FCA0 7396                    <1>             jnc    short loc_swp_retn
                                 <1> 
0000FCA2 FE05[F7650100]          <1>             inc    byte [SWP_DRV_chg]
0000FCA8 FE05[D30C0100]          <1>             inc    byte [Restore_CDIR]
0000FCAE BE[A3620100]            <1>             mov    esi, FindFile_Directory
0000FCB3 8A25[F5650100]          <1>             mov    ah, [SWP_Mode+1]
0000FCB9 E860A1FFFF              <1>             call   change_current_directory
0000FCBE 0F8274FFFFFF            <1>             jc     loc_swp_retn ; eax = error code
                                 <1> 
                                 <1> loc_swp_change_prompt_dir_string:
                                 <1>             ; esi = PATH_Array
                                 <1>             ; eax = Current Directory First Cluster
                                 <1>             ; edi = Logical DOS Drive Description Table
0000FCC4 E87AA0FFFF              <1>             call   change_prompt_dir_str
0000FCC9 29C0                    <1>             sub    eax, eax ; 0
0000FCCB E968FFFFFF              <1>             jmp    loc_swp_retn
                                 <1> 
                                 <1> reset_working_path:
                                 <1>             ; 06/10/2016 - TRDOS 386 (TRDOS v2.0)
                                 <1>             ;
                                 <1>             ; TRDOS v1.0 (DIR.ASM, "proc_reset_working_path")
                                 <1>             ; 05/02/2011 - 08/02/2011
                                 <1>             ;
                                 <1>             ; Restores current drive and directory
                                 <1>             ;
                                 <1>             ; INPUT: none
                                 <1>             ; OUTPUT: DL = SWP_DRV, EAX = 0 -> OK
                                 <1>             ;
                                 <1>             ;    AX = 0 -> ESI = Logical Dos Drv Desc. Table
                                 <1>             ;
                                 <1>             ;    EAX, EBX, ECX, EDX, ESI, EDI will be changed
                                 <1>             ;
                                 <1> 
                                 <1> 
0000FCD0 31C0                    <1>             xor    eax, eax
0000FCD2 48                      <1>             dec    eax
                                 <1> 
0000FCD3 668B15[F6650100]        <1>             mov    dx, [SWP_DRV]
0000FCDA 08F6                    <1>             or     dh, dh
0000FCDC 742E                    <1>             jz     short loc_rwp_return
                                 <1> 
0000FCDE 3A15[FE580100]          <1>             cmp    dl, [Current_Drv]
0000FCE4 7407                    <1>             je     short loc_rwp_restore_cdir
                                 <1> loc_rwp_restore_cdrv:
0000FCE6 E8D571FFFF              <1>             call   change_current_drive
0000FCEB EB10                    <1>             jmp    short loc_rwp_restore_ok
                                 <1> loc_rwp_restore_cdir:
0000FCED 31DB                    <1>             xor    ebx, ebx
0000FCEF 88D7                    <1>             mov    bh, dl
0000FCF1 BE00010900              <1>             mov    esi, Logical_DOSDisks
0000FCF6 01DE                    <1>             add    esi, ebx
                                 <1> 
0000FCF8 E87A72FFFF              <1>             call   restore_current_directory
                                 <1> 
                                 <1> loc_rwp_restore_ok:
0000FCFD 668B15[F6650100]        <1>             mov    dx, [SWP_DRV]
0000FD04 31C0                    <1>             xor    eax, eax
0000FD06 66A3[F7650100]          <1>             mov    [SWP_DRV_chg], ax
                                 <1> loc_rwp_return:
0000FD0C C3                      <1>             retn
                                 <1> 
                                 <1> get_file_name:
                                 <1>             ; 15/10/2016 - TRDOS 386 (TRDOS v2.0)
                                 <1>             ; Convert file name
                                 <1>             ;       from directory entry format
                                 <1>                 ;  to (8.3) dot file name format
                                 <1>             ;
                                 <1>             ; TRDOS v1.0 (DIR.ASM, "get_file_name")
                                 <1>                 ; 2005 - 09/10/2011
                                 <1>             ; INPUT:
```

```
                             <1>             ;      DS:SI -> Directory Entry Format File Name
                             <1>             ;        ES:DI -> DOS Dot File Name Address
                             <1>             ; OUTPUT:
                             <1>             ;      DS:SI -> DOS Dot File Name Address
                             <1>                ;  ES:DI -> Directory Entry Format File Name
                             <1>             ;
                             <1>             ; TRDOS 386 (15/10/2016)
                             <1>             ; INPUT:
                             <1>             ;      ESI = File name addr in dir entry format
                             <1>             ;      EDI = Dot file name address (destination)
                             <1>             ; OUTPUT:
                             <1>             ;      File name is converted and moved
                             <1>             ;      to destination (as 8.3 dot filename)
                             <1>             ;
                             <1>             ; Modified registers: EAX, ECX
                             <1>
                             <1>                ; 2005 (TRDOS 8086) - 2016 (TRDOS 386)
                             <1>
0000FD0D 57                  <1>             push   edi
0000FD0E 56                  <1>             push   esi
0000FD0F AC                  <1>             lodsb
0000FD10 3C20                <1>             cmp    al, 20h
0000FD12 762A                <1>             jna    short pass_gfn_ext
0000FD14 56                  <1>             push   esi
0000FD15 AA                  <1>             stosb
0000FD16 B907000000          <1>             mov    ecx, 7
                             <1> loc_gfn_next_char:
0000FD1B AC                  <1>             lodsb
0000FD1C 3C20                <1>             cmp    al, 20h
0000FD1E 7603                <1>             jna    short pass_gfn_fn
0000FD20 AA                  <1>             stosb
0000FD21 E2F8                <1>             loop   loc_gfn_next_char
                             <1> pass_gfn_fn:
0000FD23 5E                  <1>             pop    esi
0000FD24 83C607              <1>             add    esi, 7
0000FD27 AC                  <1>             lodsb
0000FD28 3C20                <1>             cmp    al, 20h
0000FD2A 7612                <1>             jna    short pass_gfn_ext
0000FD2C B42E                <1>             mov    ah, '.'
0000FD2E 86E0                <1>             xchg   ah, al
0000FD30 66AB                <1>             stosw
0000FD32 AC                  <1>             lodsb
0000FD33 3C20                <1>             cmp    al, 20h
0000FD35 7607                <1>             jna    short pass_gfn_ext
0000FD37 AA                  <1>             stosb
0000FD38 AC                  <1>             lodsb
0000FD39 3C20                <1>             cmp    al, 20h
0000FD3B 7601                <1>             jna    short pass_gfn_ext
0000FD3D AA                  <1>             stosb
                             <1> pass_gfn_ext:
0000FD3E 30C0                <1>             xor    al, al
0000FD40 AA                  <1>             stosb
0000FD41 5E                  <1>             pop    esi
0000FD42 5F                  <1>             pop    edi
0000FD43 C3                  <1>             retn
                             <1>
                             <1> set_hardware_int_vector:
                             <1>             ; 18/03/2017
                             <1>             ; 03/03/2017
                             <1>             ; 28/02/2017 - TRDOS 386 (TRDOS v2.0)
                             <1>             ;
                             <1>             ; SET/RESET HARDWARE INTERRUPT GATE
                             <1>             ;
                             <1>             ; Changes interrupt gate descriptor table
                             <1>             ; (without changing default interrupt list)
                             <1>             ;
                             <1>             ; INPUT:
                             <1>             ;      AL = IRQ number (0 to 15)
                             <1>             ;      AH > 0 -> set
                             <1>             ;      AH = 0 -> reset
                             <1>             ;
                             <1>             ; Modified registers: eax, ebx, edx, edi
                             <1>             ;
                             <1>
0000FD44 C0E002              <1>             shl    al, 2 ; IRQ number * 4
0000FD47 0FB6D8              <1>             movzx  ebx, al
                             <1>
0000FD4A 08E4                <1>             or     ah, ah
0000FD4C 7508                <1>             jnz    short shintv_1 ; set (for user call service)
                             <1>
                             <1>             ; 18/03/2017
0000FD4E 81C3[D0160100]      <1>             add    ebx, IRQ_list ; reset to default interrupt list
0000FD54 EB06                <1>             jmp    short shintv_2
                             <1> shintv_1:
0000FD56 81C3[7DFD0000]      <1>             add    ebx, IRQ_u_list
                             <1> shintv_2:
0000FD5C 8B13                <1>             mov    edx, [ebx] ; IRQ handler address
                             <1>
                             <1>             ; 03/03/2017
0000FD5E D0E0                <1>             shl    al, 1 ; IRQ number * 8
                             <1>             ; 18/03/2017
0000FD60 0FB6F8              <1>             movzx  edi, al
0000FD63 81C7[50560100]      <1>             add    edi, idt + (8*32) ; IRQ 0 offset = idt + 256
                             <1>
0000FD69 89D0                <1>             mov    eax, edx ; IRQ handler address
0000FD6B BB00000800          <1>             mov    ebx, 80000h
                             <1>
                             <1>             ;mov   edx, eax
0000FD70 66BA008E            <1>             mov    dx, 8E00h
0000FD74 6689C3              <1>             mov    bx, ax
0000FD77 89D8                <1>             mov    eax, ebx ; /* selector = 0x0008 = cs */
                             <1>                                   ; /* interrupt gate - dpl=0, present */
0000FD79 AB                  <1>             stosd  ; selector & offset bits 0-15
```

```
0000FD7A 8917                <1>             mov    [edi], edx ; attributes & offset bits 16-23
                             <1> 
0000FD7C C3                  <1>             retn
                             <1> IRQ_u_list:
                             <1>             ; 28/02/2017
0000FD7D [8B060000]          <1>             dd     timer_int
0000FD81 [FF0D0000]          <1>             dd     kb_int
0000FD85 [6D080000]          <1>             dd     irq2
0000FD89 [BDFD0000]          <1>             dd     IRQ_service3
0000FD8D [C7FD0000]          <1>             dd     IRQ_service4
0000FD91 [D1FD0000]          <1>             dd     IRQ_service5
0000FD95 [B0410000]          <1>             dd     fdc_int
0000FD99 [DBFD0000]          <1>             dd     IRQ_service7
0000FD9D [F6070000]          <1>             dd     rtc_int
0000FDA1 [E5FD0000]          <1>             dd     IRQ_service9
0000FDA5 [EFFD0000]          <1>             dd     IRQ_service10
0000FDA9 [F9FD0000]          <1>             dd     IRQ_service11
0000FDAD [03FE0000]          <1>             dd     IRQ_service12
0000FDB1 [0DFE0000]          <1>             dd     IRQ_service13
0000FDB5 [2D4B0000]          <1>             dd     hdc1_int
0000FDB9 [544B0000]          <1>             dd     hdc2_int
                             <1> 
                             <1>             ; 03/03/2017
                             <1>             ; 27/02/2017
                             <1> IRQ_service3:
0000FDBD 36C605[BA6B0100]03  <1>             mov    byte [ss:IRQnum], 3
0000FDC5 EB4E                <1>             jmp    short IRQ_service
                             <1> IRQ_service4:
0000FDC7 36C605[BA6B0100]04  <1>             mov    byte [ss:IRQnum], 4
0000FDCF EB44                <1>             jmp    short IRQ_service
                             <1> IRQ_service5:
0000FDD1 36C605[BA6B0100]05  <1>             mov    byte [ss:IRQnum], 5
0000FDD9 EB3A                <1>             jmp    short IRQ_service
                             <1> IRQ_service7:
0000FDDB 36C605[BA6B0100]07  <1>             mov    byte [ss:IRQnum], 7
0000FDE3 EB30                <1>             jmp    short IRQ_service
                             <1> IRQ_service9:
0000FDE5 36C605[BA6B0100]09  <1>             mov    byte [ss:IRQnum], 9
0000FDED EB26                <1>             jmp    short IRQ_service
                             <1> IRQ_service10:
0000FDEF 36C605[BA6B0100]0A  <1>             mov    byte [ss:IRQnum], 10
0000FDF7 EB1C                <1>             jmp    short IRQ_service
                             <1> IRQ_service11:
0000FDF9 36C605[BA6B0100]0B  <1>             mov    byte [ss:IRQnum], 11
0000FE01 EB12                <1>             jmp    short IRQ_service
                             <1> IRQ_service12:
0000FE03 36C605[BA6B0100]0C  <1>             mov    byte [ss:IRQnum], 12
0000FE0B EB08                <1>             jmp    short IRQ_service
                             <1> IRQ_service13:
0000FE0D 36C605[BA6B0100]0D  <1>             mov    byte [ss:IRQnum], 13
                             <1>             ;jmp   short IRQ_service
                             <1> IRQ_service:
                             <1>             ; 13/06/2017
                             <1>             ; 11/06/2017
                             <1>             ; 10/06/2017
                             <1>             ; 01/03/2017, 04/03/2017
                             <1>             ; 27/02/2017, 28/02/2017
0000FE15 1E                  <1>             push   ds
0000FE16 06                  <1>             push   es
0000FE17 0FA0                <1>             push   fs
0000FE19 0FA8                <1>             push   gs
                             <1> 
0000FE1B 60                  <1>             pushad ; eax,ecx,edx,ebx,esp,ebp,esi,edi
0000FE1C 66B91000            <1>             mov     cx, KDATA
0000FE20 8ED9                <1>             mov     ds, cx
0000FE22 8EC1                <1>             mov     es, cx
0000FE24 8EE1                <1>             mov     fs, cx
0000FE26 8EE9                <1>             mov     gs, cx
                             <1> 
0000FE28 0F20D8              <1>             mov    eax, cr3
0000FE2B A3[B66B0100]        <1>             mov    [IRQ_cr3], eax
                             <1> 
0000FE30 A1[38580100]        <1>             mov    eax, [k_page_dir]
0000FE35 0F22D8              <1>             mov    cr3, eax
                             <1> 
0000FE38 A0[BA6B0100]        <1>             mov    al, [IRQnum]
                             <1> 
                             <1>             ;mov   cl, [sysflg]
                             <1>             ;mov   [u.r_mode], cl  ; system (0) or user mode (FFh)
                             <1> IRQsrv_0:
0000FE3D 0FB6D8              <1>             movzx  ebx, al
0000FE40 8A9B[08160100]      <1>             mov    bl, [ebx+IRQenum] ; IRQ (available) index number + 1
                             <1>             ; 01/03/2017
0000FE46 FECB                <1>             dec    bl  ; IRQ index number, 0 to 8
0000FE48 0F8807010000        <1>             js     IRQsrv_5 ; not available to use here!?
                             <1>             ;
0000FE4E 80BB[806B0100]80    <1>             cmp    byte [ebx+IRQ.method], 80h ; using by a dev or kernel?
0000FE55 7205                <1>             jb     short IRQsrv_1 ; no
                             <1> 
                             <1>             ; If the IRQ service is already owned by TRDOS 386 kernel
                             <1>             ;       or a Device driver
                             <1>             ; we need to call 'dev_IRQ_service'
                             <1> 
                             <1>             ; IRQ number in AL
0000FE57 E868020000          <1>             call   dev_IRQ_service      ; IRQ service for device drivers
                             <1>             ; IRQ number in AL
                             <1> IRQsrv_1:
                             <1>             ; check user callback service status
                             <1>             ; AL = IRQ number
                             <1>             ; EBX = IRQ (Available) Index number
                             <1> 
0000FE5C A2[D7030300]        <1>             mov    [u.irqwait], al ; set waiting IRQ flag
                             <1> 
```

```
0000FE61 8A83[6E6B0100]      <1>                 mov    al, [ebx+IRQ.owner]
0000FE67 20C0                <1>                 and    al, al
0000FE69 0F84E6000000        <1>                 jz     IRQsrv_5 ; it is not owned by a user/proc
                             <1> 
                             <1>                 ; 03/03/2017
0000FE6F 89DA                <1>                 mov    edx, ebx
0000FE71 C0E202              <1>                 shl    dl, 2
0000FE74 8B92[926B0100]      <1>                 mov    edx, [edx+IRQ.addr] ; S.R.B. or Callback service addr
                             <1> 
0000FE7A 8AA3[806B0100]      <1>                 mov    ah, [ebx+IRQ.method]
0000FE80 F6C401              <1>                 test   ah, 1
0000FE83 7534                <1>                 jnz    short IRQsrv_4 ; Callback service method
                             <1> 
                             <1>                 ; Signal Response Byte method
                             <1>                 ;mov   edx, [edx+IRQ.addr] ; Signal Response Byte address
                             <1>                 ;                       ; (Physical address, non-swappable)
0000FE85 80E402              <1>                 and    ah, 2 ; bit 1, (S.R.B.) counter (auto increment) method
0000FE88 8AA3[896B0100]      <1>                 mov    ah, [ebx+IRQ.srb] ; Signal Response Byte value
0000FE8E 7408                <1>                 jz     short IRQsrv_2 ; fixed S.R.B. value
                             <1>                 ; counter method (auto increment)
0000FE90 FEC4                <1>                 inc    ah
0000FE92 88A3[896B0100]      <1>                 mov    [ebx+IRQ.srb], ah ; next (count) number
                             <1> IRQsrv_2:
0000FE98 8822                <1>                 mov    [edx], ah ; put S.R.B. val to the user's S.R.B. addr
0000FE9A C605[D7030300]00    <1>                 mov    byte [u.irqwait], 0 ; clear waiting IRQ flag
                             <1> 
0000FEA1 3A05[B3030300]      <1>                 cmp    al, [u.uno]
0000FEA7 0F84A8000000        <1>                 je     IRQsrv_5 ; the owner is current user/process
                             <1> IRQsrv_3:
                             <1>                 ; the owner is not current user/process
                             <1>                 ; AL = process number
0000FEAD B202                <1>                 mov    dl, 2 ; priority, 2 = event (high)
0000FEAF E837FAFFFF          <1>                 call   set_run_sequence
                             <1> 
                             <1>                 ; [u.irqwait] = waiting IRQ number for callback service
                             <1> 
0000FEB4 E99C000000          <1>                 jmp    IRQsrv_5
                             <1> IRQsrv_4:
0000FEB9 3A05[B3030300]      <1>                 cmp    al, [u.uno]  ; is the owner is current user/process?
0000FEBF 75EC                <1>                 jne    short IRQsrv_3 ; no !
                             <1> 
                             <1>                 ; Check if an IRQ callback service already in progress
0000FEC1 803D[D8030300]00    <1>                 cmp    byte [u.r_lock], 0
0000FEC8 0F8787000000        <1>                 ja     IRQsrv_5 ; nothing to do !
                             <1>                                   ; (we need to complete prev callback)
0000FECE 803D[D4030300]00    <1>                 cmp    byte [u.t_lock], 0
0000FED5 777E                <1>                 ja     short IRQsrv_5 ; nothing to do !
                             <1>                                   ; (we need to complete timer callback)
                             <1> 
                             <1>                 ; 04/03/2017
0000FED7 C605[D7030300]00    <1>                 mov    byte [u.irqwait], 0 ; reset/clear waiting IRQ flag
                             <1> 
0000FEDE FE05[D8030300]      <1>                 inc    byte [u.r_lock] ; 'IRQ callback service in progress' flag
                             <1> 
0000FEE4 8A0D[5B030300]      <1>                 mov    cl, [sysflg]   ; (system call) mode flag (kernel/user)
0000FEEA 880D[D9030300]      <1>                 mov    [u.r_mode], cl ; system mode (0) or user mode (FFh)
                             <1> 
                             <1>                 ;
0000FEF0 8B2D[D4570100]      <1>                 mov    ebp, [tss.esp0] ; kernel stack address (for ring 0)
0000FEF6 83ED14              <1>                 sub    ebp, 20             ; eip, cs, eflags, esp, ss
0000FEF9 892D[5C030300]      <1>                 mov    [u.sp], ebp
0000FEFF 8925[60030300]      <1>                 mov    [u.usp], esp
                             <1> 
                             <1>                 ;or    word [ebp+8], 200h ; 22/01/2017, force enabling interrupts
                             <1> 
0000FF05 8B44241C            <1>                 mov    eax, [esp+28] ; pushed eax
0000FF09 A3[64030300]        <1>                 mov    [u.r0], eax
                             <1> 
0000FF0E E81AE7FFFF          <1>                 call   wswap ; save user's registers & status
                             <1> 
                             <1>                 ; software int is in ring 0 but IRQ handler must return to ring 3
                             <1>                 ; so, ring 3 return address and stack registers
                             <1>                 ; (eip, cs, eflags, esp, ss)
                             <1>                 ; must be copied to IRQ handler return
                             <1>                 ; eip will be replaced by callback service routine address
                             <1> 
0000FF13 C605[5B030300]FF    <1>                 mov    byte [sysflg], 0FFh ; user mode
                             <1> 
                             <1>                 ; system mode (system call)
                             <1>                 ;mov   ebp, [u.sp] ; EIP (u), CS (UCODE), EFLAGS (u),
                             <1>                                  ; ESP (u), SS (UDATA)
                             <1> 
0000FF1A 8B4510              <1>                 mov    eax, [ebp+16]; SS (UDATA)
0000FF1D 89E6                <1>                 mov    esi, esp
0000FF1F 50                  <1>                 push   eax
0000FF20 50                  <1>                 push   eax
0000FF21 89E7                <1>                 mov    edi, esp
0000FF23 893D[60030300]      <1>                 mov    [u.usp], edi
0000FF29 B908000000          <1>                 mov    ecx, ((ESPACE/4) - 4) ; except DS, ES, FS, GS
0000FF2E F3A5                <1>                 rep    movsd
0000FF30 B104                <1>                 mov    cl, 4
0000FF32 F3AB                <1>                 rep    stosd
0000FF34 893D[5C030300]      <1>                 mov    [u.sp], edi
0000FF3A 89EE                <1>                 mov    esi, ebp
0000FF3C B105                <1>                 mov    cl, 5 ; EIP (u), CS (UCODE), EFLAGS (u), ESP (u), SS (UDATA)
0000FF3E F3A5                <1>                 rep    movsd
                             <1>                 ;
                             <1> 
0000FF40 8B0D[B8030300]      <1>                 mov    ecx, [u.pgdir]
0000FF46 890D[B66B0100]      <1>                 mov    [IRQ_cr3], ecx
                             <1> 
                             <1> set_IRQ_callback_addr:
                             <1>                 ;
                             <1>             ; This routine sets return address
                             <1>             ; to start of user's interrupt
                             <1>             ; service (callback) address
                             <1>             ;
                             <1>             ; INPUT:
                             <1>             ;     EDX = callback routine/service address
                             <1>             ;           (virtual, not physical address!)
                             <1>             ;     [u.sp] = kernel stack, points to
                             <1>             ;             user's EIP,CS,EFLAGS,ESP,SS
                             <1>             ;             registers.
                             <1>             ; OUTPUT:
                             <1>             ;     EIP (user) = callback (service) address
                             <1>             ;     CS (user) = UCODE
                             <1>             ;     EFLAGS (user) = flags before callback
                             <1>             ;      ESP (user) = ESP-4 (user, before callback)
                             <1>             ;     [ESP](user) = EIP (user) before callback
                             <1>             ;
                             <1>             ; Note: If CPU was in user mode while entering
                             <1>             ;     the timer interrupt service routine,
                             <1>             ;     'IRET' will get return to callback routine
                             <1>             ;     immediately. If CPU was in system/kernel mode
                             <1>             ;     'iret' will get return to system call and
                             <1>             ;     then, callback routine will be return address
                             <1>             ;     from system call. (User's callback/service code
                             <1>             ;     will be able to return to normal return address
                             <1>             ;     via a 'sysrele' system call at the end.)
                             <1>             ;
                             <1>             ; Note: User's IRQ callback service code must be ended
                             <1>             ;     with a 'sysrele' system call !
                             <1>             ;
                             <1>             ;     For example:
                             <1>             ;
                             <1>             ;     audio_IRQ_callback:
                             <1>             ;         ...
                             <1>             ;         <load DMA buffer with audio data>
                             <1>             ;         ...
                             <1>             ;         mov eax, 39 ; 'sysrele'
                             <1>             ;         int 40h ; TRDOS 386 system call (interrupt)
                             <1>             ;
                             <1>
                             <1>             ;mov   edx, [edx+IRQ.addr] ; Callback service address
                             <1>             ;                       ; (Virtual address)
                             <1>
0000FF4C 8B2D[5C030300]      <1>             mov    ebp, [u.sp]; kernel's stack, points to EIP (user)
0000FF52 895500              <1>             mov    [ebp], edx
                             <1> IRQsrv_5:
                             <1>             ; EOI & return
                             <1>             ; 11/06/2017
                             <1>             ; 10/06/2017
0000FF55 A0[BA6B0100]        <1>             mov    al, [IRQnum]
0000FF5A FA                  <1>             cli
0000FF5B 3C07                <1>             cmp    al, 7
0000FF5D 7604                <1>             jna    short IRQsrv_6
                             <1>             ;
                             <1>             ;mov   al, EOI      ; end of interrupt
0000FF5F B020                <1>             mov    al, 20h
                             <1>             ;cli         ; disable interrupts till stack cleared
                             <1>             ;out   INTB00, al ; For controll2 #2
0000FF61 E6A0                <1>             out    0A0h, al
                             <1> IRQsrv_6:
                             <1>             ;mov   byte [IRQnum], 0 ; reset
                             <1>             ;mov   al, EOI      ; end of interrupt
0000FF63 B020                <1>             mov    al, 20h
                             <1>             ;cli         ; disable interrupts till stack cleared
                             <1>             ;out   INTA00, al ; end of interrupt to 8259 - 1
0000FF65 E620                <1>             out    20h, al
                             <1> IRQsrv_7:
                             <1>             ;; 13/06/2017
                             <1>             ;or    word [ebp+8], 200h ; force enabling interrupts
                             <1>             ;
0000FF67 8B0D[B66B0100]      <1>             mov    ecx, [IRQ_cr3]      ; previous content of cr3 register
0000FF6D 0F22D9              <1>             mov    cr3, ecx     ; restore cr3 register content
                             <1>             ;
0000FF70 61                  <1>             popad ; edi,esi,ebp,(icrement esp by 4), ebx,edx,ecx,eax
                             <1>             ;
0000FF71 0FA9                <1>             pop    gs
0000FF73 0FA1                <1>             pop    fs
0000FF75 07                  <1>             pop    es
0000FF76 1F                  <1>             pop    ds
                             <1>             ;
0000FF77 CF                  <1>             iretd  ; return from interrupt
                             <1>
                             <1> get_device_number:
                             <1>             ; 08/10/2016
                             <1>             ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>             ;
                             <1>             ; This procedure compares name of requested
                             <1>             ; device with kernel device names and
                             <1>             ; installable device names. If names match,
                             <1>             ; the relevant device index (entry) number
                             <1>             ; will be returned the caller (sysopen)
                             <1>             ; for the requested device.
                             <1>             ;
                             <1>             ; NOTE: Installable device drivers must
                             <1>             ; be loaded before using 'sysopen'
                             <1>             ; (opendev) system call.
                             <1>             ;
                             <1>             ; INPUT:
                             <1>             ;    ESI = device name address (ASCIIZ)
                             <1>             ;         (in kernel's memory space)
                             <1>             ;    max name length = 8 without '/dev/')
                             <1>             ;    Device name will be capitalized
                             <1>             ;    and if there is, '/dev/' will be
```

```
                                <1>             ;    removed from name before comparising)
                                <1>             ;
                                <1>             ; OUTPUT:
                                <1>             ;    cf = 0 ->
                                <1>             ;      EAX (AL) = device entry/index number
                                <1>             ;    cf = 1 -> device not found (installed)
                                <1>             ;             or invalid device name
                                <1>             ;             (AL=0)
                                <1>             ;    device_name = device name address (asciiz)
                                <1>                  ;
                                <1>             ; Modified registers: EAX, EBX, ESI, EDI
                                <1>
0000FF78 BF[F9650100]           <1>             mov    edi, device_name
0000FF7D E805010000             <1>             call   lodsb_capitalize
0000FF82 88C4                   <1>             mov    ah, al
0000FF84 3C2F                   <1>             cmp    al, '/'
0000FF86 750E                   <1>             jne    short gdn_1
0000FF88 BF[F9650100]           <1>             mov    edi, device_name
0000FF8D E8F5000000             <1>             call   lodsb_capitalize
                                <1> gdn_0:
0000FF92 20C0                   <1>             and    al, al ; 0 ?
0000FF94 7420                   <1>             jz     short gdn_err ; null name after '/'
                                <1> gdn_1:
0000FF96 3C44                   <1>             cmp    al, 'D'
0000FF98 7517                   <1>             jne    short gdn_2
0000FF9A E8E8000000             <1>             call   lodsb_capitalize
0000FF9F 3C45                   <1>             cmp    al, 'E'
0000FFA1 750E                   <1>             jne    short gdn_2
0000FFA3 E8DF000000             <1>             call   lodsb_capitalize
0000FFA8 3C56                   <1>             cmp    al, 'V'
0000FFAA 7505                   <1>             jne    short gdn_2
0000FFAC AC                     <1>             lodsb
0000FFAD 3C2F                   <1>             cmp    al, '/'
0000FFAF 740D                   <1>             je     short gdn_4
                                <1> gdn_2:
0000FFB1 80FC2F                 <1>             cmp    ah, '/'
0000FFB4 750F                   <1>             jne    short gdn_5
                                <1> gdn_err:
                                <1>             ; invalid device name or device not found
0000FFB6 31C0                   <1>             xor    eax, eax ; 0
0000FFB8 F9                     <1>             stc
0000FFB9 C3                     <1>             retn
                                <1> gdn_3:
0000FFBA 3C2F                   <1>             cmp    al, '/'
0000FFBC 7507                   <1>             jne    short gdn_5
                                <1> gdn_4:
0000FFBE BF[F9650100]           <1>             mov    edi, device_name
0000FFC3 EB04                   <1>             jmp    short gdn_6
                                <1> gdn_5:
0000FFC5 3C00                   <1>             cmp    al, 0
0000FFC7 7419                   <1>             je     short gdn_7
                                <1> gdn_6:
0000FFC9 E8B9000000             <1>             call lodsb_capitalize
0000FFCE 81FF[01660100]         <1>             cmp    edi, device_name + 8
0000FFD4 72E4                   <1>             jb     short gdn_3
0000FFD6 3C00                   <1>             cmp    al, 0
0000FFD8 75DC                   <1>             jne    short gdn_err
0000FFDA 81FF[FA650100]         <1>             cmp    edi, device_name + 1
0000FFE0 76D4                   <1>             jna    short gdn_err ; null name after '/'
                                <1> gdn_7:
0000FFE2 AA                     <1>             stosb
                                <1>             ; zero padding ("NAME",0,0,0,0)
0000FFE3 81FF[01660100]         <1>             cmp    edi, device_name + 8
0000FFE9 72F7                   <1>             jb     short gdn_7
                                <1> gdn_8:
                                <1>             ; search for kernel device names
0000FFEB BE[F9650100]           <1>             mov    esi, device_name
0000FFF0 BF[EE130100]           <1>             mov    edi, KDEV_NAME
0000FFF5 31C0                   <1>             xor    eax, eax
                                <1> gdn_9:
0000FFF7 A7                     <1>             cmpsd
0000FFF8 7505                   <1>             jne    short gdn_10
0000FFFA A7                     <1>             cmpsd
0000FFFB 7503                   <1>             jne    short gdn_11
0000FFFD EB2B                   <1>             jmp    short gdn_17 ; match
                                <1> gdn_10:
0000FFFF A7                     <1>             cmpsd  ; add esi, 4 & add edi, 4
                                <1> gdn_11:
00010000 BE[F9650100]           <1>             mov    esi, device_name
00010005 FEC0                   <1>             inc    al
00010007 3C16                   <1>             cmp    al, NumOfKernelDevNames
00010009 72EC                   <1>             jb     short gdn_9
                                <1> gdn_12:
                                <1>             ; search for installable device names
                                <1>             ; esi = offset device_name
0001000B BF[24660100]           <1>             mov    edi, IDEV_NAME
00010010 28C0                   <1>             sub    al, al ; 0
                                <1> gdn_13:
00010012 A7                     <1>             cmpsd
00010013 7505                   <1>             jne    short gdn_14
00010015 A7                     <1>             cmpsd
00010016 7503                   <1>             jne    short gdn_15
00010018 EB3F                   <1>             jmp    short gdn_19 ; match
                                <1> gdn_14:
0001001A A7                     <1>             cmpsd  ; add esi, 4 & add edi, 4
                                <1> gdn_15:
0001001B BE[F9650100]           <1>             mov    esi, device_name
00010020 FEC0                   <1>             inc    al
00010022 3C08                   <1>             cmp    al, NumOfInstallableDevices
00010024 72EC                   <1>             jb     short gdn_13
                                <1>
                                <1> gdn_16:     ; error: invalid device name (not found) !
00010026 30C0                   <1>             xor    al, al
```

```
00010028 F9                   <1>             stc
00010029 C3                   <1>             retn
                              <1> 
                              <1> gdn_17:           ; name match (with one of kernel device names)
                              <1>             ;
                              <1>             ; convert KDEV_NAME index to
                              <1>             ; KDEV_NUMBER index
                              <1>             ; (different names are used for same devices)
                              <1>             ; (example: "COM1" & "TTY8" = device number 18)
0001002A 89C3                 <1>             mov    ebx, eax ; < 256
0001002C 8A83[9E140100]       <1>             mov    al, [KDEV_NUMBER+ebx]
                              <1> 
                              <1>             ; check if empty dev entry in the list
00010032 80B8[A8670100]00     <1>             cmp    byte [DEV_OPENMODE+eax], 0
00010039 771B                 <1>             ja     short gdn_18 ; it must be already set
                              <1> 
                              <1>             ; (re)set device name and access flags
                              <1>             ; (remain open work will be easy after that)
                              <1>             ; (NOTE: here, data will be copied to bss section)
0001003B 88C3                 <1>             mov    bl, al
0001003D 83EF08               <1>             sub    edi, 8 ; kernel device name address (data)
00010040 66C1E302             <1>             shl    bx, 2
00010044 89BB[C6670100]       <1>             mov    [DEV_NAME_PTR+ebx], edi ; (all) device names
0001004A 8A98[F4150100]       <1>             mov    bl, [KDEV_ACCESS+eax] ; kernel dev list (data)
00010050 8898[F4660100]       <1>             mov    [DEV_ACCESS+eax], bl ; (all) device list (bss)
                              <1> gdn_18:
00010056 FEC0                 <1>             inc    al ; 1 to NumOfKernelDevNames (<=7Fh)
                              <1>             ; eax = device index/entry number
00010058 C3                   <1>             retn
                              <1> 
                              <1> gdn_19:           ; name match (with one of installable device names)
                              <1>             ;
                              <1>             ; al = 0 to NumOfInstallableDevices - 1 (<=7Fh)
                              <1> 
00010059 89C3                 <1>             mov    ebx, eax
0001005B 80C316               <1>             add    bl, NumOfKernelDevices    ; < NUMOFDEVICES
                              <1> 
                              <1>             ; check if empty dev entry in the list
0001005E 80BB[A8670100]00     <1>             cmp    byte [DEV_OPENMODE+ebx], 0
00010065 771D                 <1>             ja     short gdn_20 ; it must be already set
                              <1> 
                              <1>             ; (re)set device name and access flags
                              <1>             ; (remain open work will be easy after that)
00010067 83EF08               <1>             sub    edi, 8 ; installable device name address
0001006A 66C1E302             <1>             shl    bx, 2 ;*4
0001006E 89BB[C6670100]       <1>             mov    [DEV_NAME_PTR+ebx], edi ; (all) device names
00010074 66C1EB02             <1>             shr    bx, 2
00010078 8A80[6C660100]       <1>             mov    al, [IDEV_FLAGS+eax] ; installable dev list
0001007E 8883[F4660100]       <1>             mov    [DEV_ACCESS+ebx], al ; (all) device list
                              <1> gdn_20:
00010084 88D8                 <1>             mov    al, bl
                              <1>             ; eax = device index/entry number ; < NUMOFDEVICES
00010086 C3                   <1>             retn
                              <1> 
                              <1> lodsb_capitalize:
                              <1>       ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
                              <1>       ; INPUT -> [esi] = character
                              <1>       ;           edi = destination
                              <1>       ; OUTPUT -> AL contains capitalized character
                              <1>       ;         esi = esi+1
                              <1>       ;         edi = edi+1
                              <1>       ;
00010087 AC                   <1>       lodsb
00010088 3C61                 <1>       cmp al, 61h
0001008A 7206                 <1>       jb  short lodsb_cap_retn
0001008C 3C7A                 <1>       cmp al, 7Ah
0001008E 7702                 <1>       ja  short lodsb_cap_retn
00010090 24DF                 <1>       and al, 0DFh
                              <1> lodsb_cap_retn:
00010092 AA                   <1>       stosb
00010093 C3                   <1>       retn
                              <1> 
                              <1> device_open:
                              <1>       ; 08/10/2016 - TRDOS 386 (TRDOS v2.0)
                              <1>       ; Complete device opening work for sysopen (device)
                              <1>       ;
                              <1>       ; INPUT ->
                              <1>       ;      EAX = Device Number (AL)
                              <1>       ;         CL = Open mode (1 = read, 2 = write)
                              <1>       ;       CH = Device access byte (bit 0 = 0)
                              <1>       ; OUTPUT ->
                              <1>       ;      EAX = Device Number
                              <1>       ;      CF = 0 -> device has been opened
                              <1>       ;      CF = 1 -> device could not be opened
                              <1>       ;
                              <1>       ;  Modified registers: ebx, (edx, ecx, esi, edi, ebp)
                              <1>       ;
                              <1> 
00010094 89C3                 <1>       mov    ebx, eax
00010096 66C1E302             <1>       shl    bx, 2 ; *4
                              <1> 
0001009A F6C580               <1>       test   ch, 80h ; bit 7, installable device driver flag
0001009D 7406                 <1>       jz     short d_open_2 ; Kernel device
                              <1>       ; installable device
                              <1> d_open_1:
0001009F FFA3[70660100]       <1>         jmp  dword [ebx+IDEV_OADDR-4]
                              <1> d_open_2:
000100A5 FFA3[B0140100]       <1>       jmp    dword [ebx+KDEV_OADDR-4]
                              <1> 
                              <1> device_close:
                              <1>       ; 08/10/2016 - TRDOS 386 (TRDOS v2.0)
                              <1>       ; Complete device closing work for sysclose (device)
                              <1>       ;
```

```
                             <1>         ; INPUT ->
                             <1>         ;      EAX = Device Number (AL)
                             <1>         ;         CL = Open mode (1 = read, 2 = write)
                             <1>         ;       CH = Device access byte (bit 0 = 0)
                             <1>         ; OUTPUT ->
                             <1>         ;      EAX = Device Number
                             <1>         ;      CF = 0 -> device has been closed
                             <1>         ;      CF = 1 -> device could not be closed
                             <1>         ;
                             <1>         ; Modified registers: ebx, (edx, ecx, esi, edi, ebp)
                             <1>         ;
                             <1>
000100AB 89C3                <1>         mov    ebx, eax
000100AD 66C1E302            <1>         shl    bx, 2 ; *4
                             <1>
000100B1 F6C580              <1>         test   ch, 80h ; bit 7, installable device driver flag
000100B4 7406                <1>         jz     short d_close_2 ; Kernel device
                             <1>         ; installable device
                             <1> d_close_1:
000100B6 FFA3[90660100]      <1>           jmp  dword [ebx+IDEV_CADDR-4]
                             <1> d_close_2:
000100BC FFA3[00150100]      <1>         jmp    dword [ebx+KDEV_CADDR-4]
                             <1>
                             <1> rnull:
                             <1>         ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>         ; read null (read from null device)
000100C2 C3                  <1>         retn
                             <1>
                             <1> wnull:
                             <1>         ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>         ; write null (write to null device)
000100C3 C3                  <1>         retn
                             <1>
                             <1> dev_IRQ_service:
                             <1>         ; 12/05/2017
                             <1>         ; 13/04/2017
                             <1>         ; 27/02/2017 - TRDOS 386 (TRDOS v2.0)
                             <1>         ; INPUT ->
                             <1>         ;      AL = IRQ Number (0 to 15)
                             <1>         ;
000100C4 53                  <1>         push   ebx
000100C5 0FB6D8              <1>         movzx  ebx, al
000100C8 C0E302              <1>         shl    bl, 2 ; * 4
000100CB 8B9B[2E6B0100]      <1>         mov    ebx, [ebx+DEV_INT_HNDLR]
000100D1 21DB                <1>         and    ebx, ebx
000100D3 7404                <1>           jz   short dIRQ_s_retn
000100D5 50                  <1>         push   eax
                             <1>
000100D6 FFD3                <1>         call   ebx
                             <1>
000100D8 58                  <1>         pop    eax
                             <1> dIRQ_s_retn:
000100D9 5B                  <1>         pop    ebx
000100DA C3                  <1>         retn
                             <1>
                             <1>
                             <1> set_dev_IRQ_service:
                             <1>         ; 13/04/2017 - TRDOS 386 (TRDOS v2.0)
                             <1>         ;
                             <1>         ; Set Device Interrupt Service
                             <1>         ;
                             <1>         ; INPUT ->
                             <1>         ;      AL = IRQ Number
                             <1>         ;      EBX = Hardware Interrupt Service Address
                             <1>         ;
                             <1>         ; Note: There is not a validation check here
                             <1>         ;      because this procedure is called by
                             <1>         ;      TRDOS 386 kernel !
                             <1>         ;        (Even if a device driver does not exist
                             <1>         ;      this setting may be used by sysaudio
                             <1>         ;      and other system calls for hardware
                             <1>         ;      components which use IRQ method for I/O.)
                             <1>         ;
                             <1>         ;push  esi
000100DB 0FB6F0              <1>         movzx  esi, al
000100DE 66C1E602            <1>         shl    si, 2 ; * 4
000100E2 899E[2E6B0100]      <1>         mov    [esi+DEV_INT_HNDLR], ebx
                             <1>         ;pop   esi
000100E8 C3                  <1>         retn
                             <1>
                             <1>
                             <1> sysaudio: ; AUDIO FUNCTIONS
                             <1>         ; 10/10/2017
                             <1>         ; 22/06/2017
                             <1>         ; 28/05/2017, 04/06/2017, 05/06/2017, 10/06/2017
                             <1>         ; 01/05/2017, 12/05/2017, 15/05/2017, 20/05/2017
                             <1>         ; 21/04/2017, 22/04/2017, 23/04/2017, 24/04/2017
                             <1>         ; 10/04/2017, 13/04/2017, 14/04/2017, 16/04/2017
                             <1>         ; 03/04/2017 (VIA VT8237R)
                             <1>         ; 01/04/2016 (trdosk6.s -> tdosk8.s)
                             <1>         ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>         ;
                             <1>         ; Inputs:
                             <1>         ;
                             <1>         ;      BH = 0 -> Beep (PC Speaker)
                             <1>         ;           BL = Duration Counter (1 for 1/64 second)
                             <1>         ;           CX = Frequency Divisor (1193180/Frequency)
                             <1>         ;              (1331 for 886 Hz)
                             <1>         ;
                             <1>         ;      01/04/2017
                             <1>         ;
                             <1>         ;      BH = 1 -> DETECT (& ENABLE) AUDIO DEVICE
                             <1>         ;           BL = 0 : PC SPEAKER
```

```
<1>       ;              1 : SOUND BLASTER 16
<1>       ;              2 : INTEL AC'97
<1>       ;              3 : VIA VT8237R (VT8233)
<1>       ;              4 : INTEL HDA
<1>       ;           5-FEh : unknown/invalid
<1>       ;             ; 04/06/2017
<1>       ;            FFh : Get current audio device id
<1>       ;
<1>       ;     BH = 2 -> ALLOCATE AUDIO BUFFER (for user)
<1>       ;            ECX = Audio Buffer Size (must be equal to
<1>       ;                  the half of DMA buffer size)
<1>       ;            EDX = Virtual Address of the buffer
<1>       ;                  (This is not DMA buffer!)
<1>       ;
<1>       ;     BH = 3 -> INITIALIZE AUDIO DEVICE
<1>       ;          BL = 0,2 -> for Signal Response Byte
<1>       ;            CL = Signal Response Byte Value (fixed)
<1>       ;                         if BL = 0
<1>       ;                  auto increment of S.R.B. value
<1>       ;                         if BL = 2
<1>       ;             EDX = Signal Response (Return) Byte Address
<1>       ;
<1>       ;          BL = 1 for CallBack Method
<1>       ;            EDX = CallBack Service Address (Virtual)
<1>       ;
<1>       ;          BL > 2 -> invalid function
<1>       ;
<1>       ;         (Audio buffer must be allocated before
<1>       ;          initialization.)
<1>       ;
<1>       ;     BH = 4 -> START TO PLAY
<1>       ;          BL = Mode
<1>       ;              Bit 0 = mono/stereo (1 = stereo)
<1>       ;              Bit 1 = 8 bit / 16 bit (1 = 16 bit)
<1>       ;          CX = Sampling Rate (Hz)
<1>       ;
<1>       ;     BH = 5 -> PAUSE
<1>       ;          BL = Any
<1>       ;
<1>       ;     BH = 6 -> CONTINUE TO PLAY
<1>       ;          BL = Any
<1>       ;
<1>       ;     BH = 7 -> STOP
<1>       ;          BL = Any
<1>       ;
<1>       ;     BH = 8 -> RESET
<1>       ;          BL = Any
<1>       ;
<1>       ;     BH = 9 -> CANCEL (CALLBACK or S.R.B. SERVICE)
<1>       ;          BL = Any
<1>       ;
<1>       ;     BH = 10 -> DEALLOCATE AUDIO BUFFER (for user)
<1>       ;          BL = Any
<1>       ;
<1>       ;     BH = 11 -> SET VOLUME LEVEL
<1>       ;          BL: (Bit 0 to 6)
<1>       ;              0 = Master (Playback, Lineout) volume
<1>       ;          CL = Left Channel Volume
<1>       ;          CH = Right Channel Volume
<1>       ;
<1>       ;          Note: If BL >= 80h (Bit 7 of BL is set),
<1>       ;          volume level will be set for next playing
<1>       ;          (actual volume level will not be changed
<1>       ;          immediately)
<1>       ;
<1>       ;     BH = 12 -> DISABLE AUDIO DEVICE
<1>       ;          (reset audio device and unlink dma buffer)
<1>       ;          BL = Any
<1>       ;
<1>       ;     12/05/2017
<1>       ;     BH = 13 -> MAP DMA BUFFER TO USER
<1>       ;         (for direct access to system's dma buffer)
<1>       ;
<1>       ;          ECX = map size in bytes
<1>       ;              (will be rounded up to page borders)
<1>       ;          EDX = Virtual Address of the buffer
<1>       ;              (Will be rounded up to page borders)
<1>       ;
<1>       ;     05/06/2017
<1>       ;     04/06/2017
<1>       ;     BH = 14 -> GET AUDIO DEVICE INFO
<1>       ;          BL: 0 = Audio Controller Info
<1>       ;            > 0 = Invalid for now!
<1>       ;
<1>       ;     22/06/2017
<1>       ;     BH = 15 -> GET CURRENT SOUND DATA (for graphics)
<1>       ;          BL: 0 -> PCM OUT data
<1>       ;            > 0 -> Invalid for now!
<1>       ;          ECX = 0 -> Get DMA Buffer Pointer
<1>       ;             EDX = Not Used
<1>       ;          ECX > 0 -> Byte count for buffer (EDX)
<1>       ;              EDX = Buffer Address (Virtual)
<1>       ;
<1>       ;     10/10/2017
<1>       ;     BH = 16 -> UPDATE DMA BUFFER DATA
<1>       ;               (by using the Audio Buffer content)
<1>       ;          BL = 0 : Update dma half buffer in sequence
<1>       ;                  (automatic destination)
<1>       ;              1 : Update 1st half of the dma buffer
<1>       ;              2 : Update 2nd half of the dma buffer
<1>       ;              3-FEh: Invalid!
<1>       ;              FFh = Get current flag value
<1>       ;                  (Half buffer number -1)
```

```
                             <1>       ;
                             <1>       ;
                             <1>       ; Outputs:
                             <1>       ;
                             <1>       ;     For BH = 0 -> Beep
                             <1>       ;         None
                             <1>       ;
                             <1>       ;     01/04/2017
                             <1>       ;
                             <1>       ;     For BH = 1 -> DETECT (& ENABLE) AUDIO DEVICE
                             <1>       ;         AH = 0 : PC SPEAKER
                             <1>       ;             1 : SOUND BLASTER 16
                             <1>       ;             2 : INTEL AC'97
                             <1>       ;             3 : VIA VT8237R (VT8233)
                             <1>       ;             4 : INTEL HDA
                             <1>       ;           5-FFh : unknown/invalid
                             <1>       ;         AL = mode status
                             <1>       ;             bit 0 = mono /stereo (1 = stereo)
                             <1>       ;             bit 1 = 8 bit / 16 bit ( 1 = 16 bit)
                             <1>       ;         04/06/2017
                             <1>       ;         EBX = PCI DEVICE/VENDOR ID (if >0)
                             <1>       ;             (BX = VENDOR ID)
                             <1>       ;         (if CF = 1 -> Error code in EAX)
                             <1>       ;
                             <1>       ;     For BH = 2 -> ALLOCATE AUDIO BUFFER (for user)
                             <1>       ;         EAX = Physical Address of the buffer
                             <1>       ;         (if CF = 1 -> Error code in EAX)
                             <1>       ;
                             <1>       ;     For BH = 3 -> INITIALIZE AUDIO DEVICE
                             <1>       ;         (if CF = 1 -> Error code in EAX)
                             <1>       ;
                             <1>       ;     For BH = 4 -> START TO PLAY
                             <1>       ;         none (if CF = 1 -> Error code in EAX)
                             <1>       ;
                             <1>       ;     For BH = 5 -> PAUSE
                             <1>       ;         none (if CF = 1 -> Error code in EAX)
                             <1>       ;
                             <1>       ;     For BH = 6 -> CONTINUE TO PLAY
                             <1>       ;         none (if CF = 1 -> Error code in EAX)
                             <1>       ;
                             <1>       ;     For BH = 7 -> STOP
                             <1>       ;         none (if CF = 1 -> Error code in EAX)
                             <1>       ;
                             <1>       ;     For BH = 8 -> RESET
                             <1>       ;         none (if CF = 1 -> Error code in EAX)
                             <1>       ;
                             <1>       ;     For BH = 9 -> CANCEL (CALLBACK or S.R.B. SERVICE)
                             <1>       ;         none (if CF = 1 -> Error code in EAX)
                             <1>       ;
                             <1>       ;     For BH = 10 -> DEALLOCATE AUDIO BUFFER (for user)
                             <1>       ;         none (if CF = 1 -> Error code in EAX)
                             <1>       ;
                             <1>       ;     For BH = 11 -> SET VOLUME LEVEL
                             <1>       ;         none (if CF = 1 -> Error code in EAX)
                             <1>       ;
                             <1>       ;     For BH = 12 -> DISABLE AUDIO DEVICE
                             <1>       ;         none (if CF = 1 -> Error code in EAX)
                             <1>       ;
                             <1>       ;     12/05/2017
                             <1>       ;     For BH = 13 -> MAP DMA BUFFER TO USER
                             <1>       ;         EAX = Physical Address of the buffer
                             <1>       ;         (if CF = 1 -> Error code in EAX)
                             <1>       ;
                             <1>       ;     04/06/2017
                             <1>       ;     For BH = 14 -> GET AUDIO DEVICE INFO
                             <1>       ;     (for BL = 0) ; 05/06/2017
                             <1>       ;         EAX = IRQ Number in AL
                             <1>       ;              Audio Device Number in AH
                             <1>       ;         EBX = DEV/VENDOR ID
                             <1>       ;             (DDDDDDDDDDDDDDDDVVVVVVVVVVVVVVVV)
                             <1>       ;         ECX = BUS/DEV/FN
                             <1>       ;             (00000000BBBBBBBBDDDDDFFF00000000)
                             <1>       ;         EDX = NABMBAR/NAMBAR (for AC97)
                             <1>       ;              (Low word, DX = NAMBAR address)
                             <1>       ;         EDX = Base IO Addr (DX) for SB16 & VT8233
                             <1>       ;         (if CF = 1 -> Error code in EAX)
                             <1>       ;                      (ERR_DEV_NOT_RDY = 15)
                             <1>       ;
                             <1>       ;     22/06/2017
                             <1>       ;     For BH = 15 -> GET CURRENT SOUND DATA
                             <1>       ;                    (for graphics)
                             <1>       ;     (for BL = 0)
                             <1>       ;      If ECX input is 0
                             <1>       ;         EAX = DMA Buffer Current Position (Offset)
                             <1>       ;      If ECX input >  0
                             <1>       ;         EAX = Actual transfer count
                             <1>       ;         (Sound samples will be copied from
                             <1>       ;          Current DMA Buffer Position to EDX
                             <1>       ;          virtual address as EAX bytes.)
                             <1>       ;      ((If CF = 1 ->  Error code in EAX))
                             <1>       ;
                             <1>       ;
                             <1>       ;     10/10/2017
                             <1>       ;     For BH = 16 -> UPDATE DMA BUFFER DATA
                             <1>       ;         EAX = 0, if the updated (or current)
                             <1>       ;                 half buffer is DMA half buffer 1
                             <1>       ;         EAX = 1, if the updated (or current)
                             <1>       ;                 half buffer is DMA half buffer 2
                             <1>       ;         (If CF = 1 -> Error code in EAX)
                             <1>       ;
                             <1>
000100E9 80FF11              <1>       cmp    bh, AUDIO1L/4
000100EC 0F83ECC5FFFF        <1>       jnb    sysret
```

```
                             <1> 
000100F2 C0E702              <1>       shl    bh, 2 ; *4
000100F5 0FB6F7              <1>       movzx esi, bh
                             <1> 
                             <1>       ; 22/04/2017
000100F8 31C0                <1>       xor    eax, eax
000100FA A3[64030300]        <1>       mov    [u.r0], eax  ; 0
                             <1> 
000100FF FF96[0A010100]      <1>       call   dword [esi+AUDIO1]
                             <1>       ;jc    error
00010105 E9D4C5FFFF          <1>       jmp    sysret
                             <1> 
0001010A [A11D0000]          <1> AUDIO1:      dd     beep ; FUNCTION = 0 (bl = Duration Counter
                             <1>                  ;                   cx = Frequency Divisor
0001010E [4E010100]          <1>       dd     soundc_detect
00010112 [EA010100]          <1>       dd     sound_alloc
00010116 [A1020100]          <1>       dd     soundc_init
0001011A [59040100]          <1>       dd     sound_play
0001011E [EF040100]          <1>       dd     sound_pause
00010122 [19050100]          <1>       dd     sound_continue
00010126 [43050100]          <1>       dd     sound_stop
0001012A [6C050100]          <1>       dd     soundc_reset
0001012E [9D050100]          <1>       dd     soundc_cancel
00010132 [C3050100]          <1>       dd     sound_dalloc
00010136 [EE050100]          <1>       dd     sound_volume
0001013A [40060100]          <1>       dd     soundc_disable
0001013E [B2060100]          <1>       dd     sound_dma_map
00010142 [21070100]          <1>       dd     soundc_info
00010146 [80070100]          <1>       dd     sound_data
0001014A [2D080100]          <1>       dd     sound_update
                             <1> 
                             <1> AUDIO1L      EQU    $ - AUDIO1
                             <1> 
                             <1> soundc_detect:
                             <1>       ; FUNCTION = 1
                             <1>       ; bl = Audio device type number
                             <1>       ; (0= pc speaker, 1 = sound blaster 16, 2 = intel ac97
                             <1>       ;  3= via vt823x, 4 = intel HDA, 0FFh= any)
                             <1> 
                             <1>       ; 04/06/2017
0001014E 8A25[BD6B0100]      <1>       mov    ah, [audio_device]
00010154 80FBFF              <1>       cmp    bl, 0FFh ; get current audio device id
00010157 7408                <1>       je     short sysaudio0
                             <1> 
00010159 20E4                <1>       and    ah, ah
0001015B 741E                <1>       jz     short soundc_get_dev
                             <1> 
0001015D 38DC                <1>       cmp    ah, bl
0001015F 7567                <1>       jne    short soundc_dev_err
                             <1> 
                             <1> sysaudio0:
00010161 A0[BE6B0100]        <1>       mov    al, [audio_mode]
                             <1> sysaudio1:
00010166 A3[64030300]        <1>       mov    [u.r0], eax
0001016B 8B1D[C86B0100]      <1>       mov    ebx, [audio_vendor] ; (DEVICE/VENDOR ID)
00010171 8B2D[60030300]      <1>       mov    ebp, [u.usp]
00010177 895D10              <1>       mov    [ebp+16], ebx  ; ebx
0001017A C3                  <1>       retn
                             <1> 
                             <1> soundc_get_dev:
                             <1>       ; 28/05/2017
                             <1>       ; 03/04/2017, 24/04/2017
0001017B C605[BC6B0100]00    <1>       mov    byte [audio_pci], 0
00010182 80FB03              <1>       cmp    bl, 3 ; VIA VT8233 (VT8237R) Audio Controller & AC97 Codec
                             <1>       ;jne   short soundc_get_dev_sb
                             <1>       ; 28/05/2017
00010185 7220                <1>       jb     short soundc_get_dev_sb
00010187 773F                <1>       ja     short soundc_dev_err   ; temporary (28/05/2017)
                             <1>       ;
00010189 E83A180000          <1>       call   DetectVT8233
0001018E 7238                <1>       jc     short soundc_dev_err
                             <1>       ; eax = 0
                             <1> 
                             <1>       ;mov   ebx, [audio_vendor]
                             <1>       ; ebx = DEVICE/VENDOR ID
                             <1>       ;      DDDDDDDDDDDDDDDDVVVVVVVVVVVVVVVV
                             <1> 
00010190 B003                <1>       mov    al, 3  ; VIA VT8237R (VT3233) Audio Controller
00010192 88C4                <1>       mov    ah, al
                             <1> 
                             <1> soundc_get_pci_dev_ok: ; 28/05/2017
00010194 FE05[BC6B0100]      <1>       inc    byte [audio_pci] ; = 1
                             <1> soundc_get_dev_ok:
                             <1> 
                             <1> soundc_get_dev_sb16_ok:
0001019A A2[BD6B0100]        <1>       mov    [audio_device], al
0001019F 8825[BE6B0100]      <1>       mov    [audio_mode], ah ; stereo (bit0), 16 bit (bit1) capability
000101A5 EBBF                <1>       jmp    short sysaudio1
                             <1> 
                             <1> soundc_get_dev_sb:
                             <1>       ; 24/04/2017
000101A7 80FB01              <1>       cmp    bl, 1 ; Sound Blaster 16
000101AA 750E                <1>       jne    short soundc_get_dev_ich ; 28/05/2017
                             <1>       ;
000101AC E8451D0000          <1>       call   DetectSB
000101B1 7215                <1>       jc     short soundc_dev_err
000101B3 B801030000          <1>       mov    eax, 0301h ; Sound Blaster 16
000101B8 EBE0                <1>       jmp    short soundc_get_dev_sb16_ok
                             <1> 
                             <1> soundc_get_dev_ich:
                             <1>       ; 28/05/2017
                             <1>       ;cmp   bl, 2 ; Intel AC'97 Audio Controller (ICH)
                             <1>       ;jne   short soundc_dev_err ; Temporary  (28/05/2017)
                             <1>       ;                        ; (Here will be modified just after
                             <1>       ;                        ; new sound card code will be ready!)
000101BA E8FC170000          <1>       call   DetectICH
000101BF 7207                <1>       jc     short soundc_dev_err
                             <1>       ;
000101C1 B802030000          <1>       mov    eax, 0302h ; AC'97 (ICH)
000101C6 EBCC                <1>       jmp    short soundc_get_pci_dev_ok
                             <1>
                             <1> soundc_dev_err:
000101C8 B80F000000          <1>       mov    eax, ERR_DEV_NOT_RDY ; Device not ready !
000101CD EB0C                <1>       jmp    short sysaudio_err
                             <1>
                             <1> sound_buff_error:
000101CF B82E000000          <1>       mov    eax, ERR_BUFFER ; Buffer error !
000101D4 EB05                <1>       jmp    short sysaudio_err
                             <1>
                             <1> soundc_respond_err:
                             <1>       ; ERR_TIME_OUT ; 'time out !' error
000101D6 B819000000          <1>       mov    eax, ERR_DEV_NOT_RESP ; 'device not responding !' error
                             <1> sysaudio_err:
000101DB A3[64030300]        <1>       mov    [u.r0], eax
000101E0 A3[C8030300]        <1>       mov    [u.error], eax
000101E5 E9D4C4FFFF          <1>       jmp    error
                             <1>
                             <1> sound_alloc:
                             <1>       ; FUNCTION =  2
                             <1>       ; ecx = audio buffer size (in bytes)
                             <1>       ; edx = audio buffer address (virtual)
                             <1>       ; 28/05/2017
                             <1>       ; 01/05/2017, 15/05/2017
                             <1>       ; 21/04/2017, 24/04/2017
000101EA 803D[BC6B0100]00    <1>       cmp    byte [audio_pci], 0
000101F1 7708                <1>       ja     short snd_alloc_0
                             <1>       ; Max. 64KB DMA buffer !!!
000101F3 81F900800000        <1>       cmp    ecx, 32768
000101F9 77D4                <1>       ja     short sound_buff_error
                             <1> snd_alloc_0:
                             <1>       ; 15/05/2017
000101FB 81F900100000        <1>       cmp    ecx, 4096 ; PAGE_SIZE
00010201 72CC                <1>       jb     short sound_buff_error
                             <1>       ;
00010203 A1[D06B0100]        <1>       mov    eax, [audio_buffer] ; audio buffer address (current)
00010208 09C0                <1>       or     eax, eax
0001020A 7445                <1>       jz     short snd_alloc_2
                             <1>       ; audio buffer exists !
0001020C 8A1D[B3030300]      <1>       mov    bl, [u.uno]
00010212 3A1D[E56B0100]      <1>       cmp    bl, [audio_user]
00010218 0F85F5000000        <1>       jne    sndc_owner_error ; not owner !
0001021E 39D0                <1>       cmp    eax, edx ; same virtual buffer address ?
00010220 7508                <1>       jne    short snd_alloc_1
00010222 3B0D[D86B0100]      <1>       cmp    ecx, [audio_buff_size]
00010228 746C                <1>       je     short snd_alloc_3 ; Nothing to do !
                             <1>                              ; Buffer has been set already!
                             <1> snd_alloc_1:
0001022A 51                  <1>       push   ecx
0001022B 52                  <1>       push   edx
0001022C 89C3                <1>       mov    ebx, eax ; audio buffer address (current)
0001022E 8B0D[D86B0100]      <1>       mov    ecx, [audio_buff_size]
00010234 E84655FFFF          <1>       call   deallocate_user_pages
00010239 5A                  <1>       pop    edx
0001023A 59                  <1>       pop    ecx
0001023B 31C0                <1>       xor    eax, eax ; 0
0001023D A3[D06B0100]        <1>       mov    [audio_buffer], eax  ; 0
00010242 A3[D46B0100]        <1>       mov    [audio_p_buffer], eax  ; 0
00010247 A3[D86B0100]        <1>       mov    [audio_buff_size], eax
0001024C A2[E56B0100]        <1>       mov    [audio_user], al ; 0
                             <1> snd_alloc_2:
00010251 89D3                <1>       mov    ebx, edx
                             <1>       ; 01/05/2017
00010253 BA00F0FFFF          <1>       mov    edx, ~PAGE_OFF ; truncating page offsets
                             <1>                            ; for aligning to page borders
                             <1>       ;and   eax, edx
00010258 21D3                <1>       and    ebx, edx
0001025A 21D1                <1>       and    ecx, edx
                             <1>       ; 15/05/2017
                             <1>       ; EAX = Beginning address (physical)
                             <1>       ; EAX = 0 -> Allocate mem block from the 1st proper aperture
                             <1>       ; ECX = Number of bytes to be allocated
0001025C E8C351FFFF          <1>       call   allocate_memory_block
00010261 0F8268FFFFFF        <1>       jc     sound_buff_error
                             <1>       ; EAX = Physical address of the allocated memory block
                             <1>       ; ECX = Allocated bytes (as truncated to page border)
                             <1>       ; EBX = Virtual address (as truncated to page border)
00010267 50                  <1>       push   eax
00010268 53                  <1>       push   ebx
00010269 51                  <1>       push   ecx
0001026A E80556FFFF          <1>       call   allocate_user_pages
0001026F 59                  <1>       pop    ecx
00010270 5B                  <1>       pop    ebx
00010271 58                  <1>       pop    eax
00010272 7223                <1>       jc     short snd_alloc_4  ; insufficient memory, buff error
                             <1>       ; eax = physical address of the user's audio buffer
                             <1>       ; ebx = virtual address of the user's audio buffer
                             <1>       ; ecx = buffer size (in bytes)
00010274 A3[D46B0100]        <1>       mov    [audio_p_buffer], eax
00010279 891D[D06B0100]      <1>       mov    [audio_buffer], ebx
0001027F 890D[D86B0100]      <1>       mov    [audio_buff_size], ecx
00010285 8A15[B3030300]      <1>       mov    dl, [u.uno]
0001028B 8815[E56B0100]      <1>       mov    [audio_user], dl
00010291 A3[64030300]        <1>       mov    [u.r0], eax
                             <1> snd_alloc_3:
00010296 C3                  <1>       retn
                             <1> snd_alloc_4:
                             <1>       ; 15/05/2017
                             <1>       ; EAX = Beginning address (physical)
                             <1>       ; ECX = Number of bytes to be deallocated
00010297 E89553FFFF           <1>       call   deallocate_memory_block
0001029C E92EFFFFFF           <1>       jmp    sound_buff_error  ; insufficient memory, buff error
                             <1>
                             <1> soundc_init:
                             <1>       ; FUNCTION = 3
                             <1>       ; bl = method (0= s.r.b., 1= callback, 2= auto incr s.r.b.)
                             <1>       ; cl = signal response byte (initial or fixed) value
                             <1>       ; edx = signal response byte or callback address
                             <1>       ; 28/05/2017
                             <1>       ; 12/05/2017, 20/05/2017
                             <1>       ; 22/04/2017, 23/04/2017, 24/04/2017
                             <1>       ; 13/04/2017, 14/04/2017, 16/04/2017, 21/04/2017
                             <1>       ; 03/04/2017, 10/04/2017
                             <1>
000102A1 A0[BD6B0100]         <1>       mov    al, [audio_device]
000102A6 20C0                 <1>       and    al, al
000102A8 7549                 <1>       jnz    short sndc_init_6
                             <1>       ;
000102AA C605[BC6B0100]00     <1>       mov    byte [audio_pci], 0
000102B1 52                   <1>       push   edx
000102B2 53                   <1>       push   ebx
000102B3 51                   <1>       push   ecx
000102B4 E83D1C0000           <1>       call   DetectSB
000102B9 7213                 <1>       jc     short sndc_init_8
000102BB 66B80103             <1>       mov    ax, 0301h ; Sound Blaster 16
000102BF EB1E                 <1>       jmp    short sndc_init_7
                             <1>
                             <1> sndc_init_11:
                             <1>       ; 28/05/2017
000102C1 E8F5160000           <1>       call   DetectICH ; Detect AC'97 (ICH) Audio Controller
000102C6 7217                 <1>       jc     short sndc_init_7
000102C8 66B80203             <1>       mov    ax, 0302h ; Intel AC'97 Audio Device
000102CC EB0B                 <1>       jmp    short sndc_init_12 ; (PCI device)
                             <1>
                             <1> sndc_init_8:
000102CE E8F5160000           <1>       call   DetectVT8233
                             <1>       ;jc    short sndc_init_7
000102D3 72EC                 <1>       jc     sndc_init_11 ; 28/05/2017
                             <1>       ; eax = 0
000102D5 B003                 <1>       mov    al, 3  ; VIA VT8237R (VT3233) Audio Controller
000102D7 88C4                 <1>       mov    ah, al
                             <1>
                             <1> sndc_init_12:
000102D9 FE05[BC6B0100]       <1>       inc    byte [audio_pci] ; = 1
                             <1> sndc_init_7:
000102DF 59                   <1>       pop    ecx
000102E0 5B                   <1>       pop    ebx
000102E1 5A                   <1>       pop    edx
000102E2 0F82E0FEFFFF         <1>       jc     soundc_dev_err
                             <1>       ;
000102E8 A2[BD6B0100]         <1>       mov    [audio_device], al
000102ED 8825[BE6B0100]       <1>       mov    [audio_mode], ah ; stereo (bit0), 16 bit (bit1) capability
                             <1>
                             <1> sndc_init_6:
000102F3 833D[D06B0100]00     <1>       cmp    dword [audio_buffer], 0
000102FA 0F86CFFEFFFF         <1>       jna    sound_buff_error
                             <1>
00010300 A0[B3030300]         <1>       mov    al, [u.uno]
00010305 8A25[E56B0100]       <1>       mov    ah, [audio_user]
0001030B 08E4                 <1>       or     ah, ah
0001030D 7418                 <1>       jz     short sndc_init0
0001030F 38E0                 <1>       cmp    al, ah
00010311 7419                 <1>       je     short sndc_init1
                             <1>
                             <1> sndc_owner_error:
00010313 B80B000000           <1>       mov    eax, ERR_NOT_OWNER ; 'permission denied !' error
                             <1> sndc_perm_error:
00010318 A3[64030300]         <1>       mov    [u.r0], eax
0001031D A3[C8030300]         <1>       mov    [u.error], eax
00010322 E997C3FFFF           <1>       jmp    error
                             <1> sndc_init0:
00010327 A2[E56B0100]         <1>       mov    [audio_user], al
                             <1> sndc_init1:
0001032C 8915[E86B0100]       <1>       mov    [audio_cb_addr], edx
00010332 881D[E66B0100]       <1>       mov    [audio_cb_mode], bl
00010338 880D[E76B0100]       <1>       mov    [audio_srb], cl
                             <1>
                             <1>       ; 24/04/2017
0001033E 803D[BD6B0100]03     <1>       cmp    byte [audio_device], 3 ; VT8233 (VT8237R)
00010345 7438                 <1>       je     short sndc_init_9
                             <1>       ;ja    short soundc_respond_err ; temporary (28/05/2017)
00010347 803D[BD6B0100]01     <1>       cmp    byte [audio_device], 1 ; SB 16
0001034E 7510                 <1>       jne    short sndc_init_13
00010350 BB[1B210100]         <1>       mov    ebx, sb16_int_handler
                             <1>       ; Note: 'SbInit' is at 'Start to Play' stage
                             <1>       ; 20/05/2017
00010355 66C705[F26B0100]08-  <1>       mov    word [audio_master_volume], 0808h ; 2/8
0001035D 08                   <1>
0001035E EB3F                 <1>       jmp    short sndc_init_10
                             <1> sndc_init_13:
                             <1>       ; 28/05/2017
00010360 803D[BD6B0100]02     <1>       cmp    byte [audio_device], 2 ; AC 97 (ICH)
00010367 0F8569FEFFFF         <1>       jne    soundc_respond_err ; temporary (28/05/2017)
                             <1>
0001036D E8FE1E0000           <1>       call   ac97_codec_config
00010372 0F825EFEFFFF         <1>       jc     soundc_respond_err ; codec error !
                             <1>
00010378 BB[57240100]         <1>       mov    ebx, ac97_int_handler
0001037D EB20                 <1>       jmp    short sndc_init_10
                             <1>
                             <1> sndc_init_9:
                             <1>       ;call  reset_codec
                             <1>       ;; eax = 1
                             <1>       ;call  codec_io_w16 ; w32
0001037F E8BB170000           <1>       call   init_codec ; 28/05/2017
00010384 0F824CFEFFFF         <1>       jc     soundc_respond_err ; codec error !
                             <1>
0001038A E8EC190000           <1>       call   channel_reset
                             <1>
                             <1>       ; setup the Codec (actually mixer registers)
0001038F E8F6180000           <1>         call    codec_config  ; unmute codec, set rates.
00010394 0F823CFEFFFF         <1>       jc     soundc_respond_err ; codec error !
                             <1>
0001039A BB[F71C0100]         <1>       mov    ebx, vt8233_int_handler
                             <1> sndc_init_10:
                             <1>       ; 13/04/2017
0001039F A0[BF6B0100]         <1>       mov    al, [audio_intr] ; IRQ number
000103A4 E832FDFFFF           <1>       call   set_dev_IRQ_service
                             <1>
                             <1>       ; SETUP (audio) INTERRUPT CALLBACK SERVICE
000103A9 8A1D[BF6B0100]       <1>       mov    bl, [audio_intr] ; IRQ number
000103AF 8A3D[E66B0100]       <1>       mov    bh, [audio_cb_mode]
000103B5 FEC7                 <1>       inc    bh   ; 1 = Signal Response Byte method (fixed value)
                             <1>                 ; 2 = Callback service method
                             <1>                 ; 3 = Auto Increment S.R.B. method
000103B7 8A0D[E76B0100]       <1>       mov    cl, [audio_srb]
000103BD 8B15[E86B0100]       <1>       mov    edx, [audio_cb_addr]
000103C3 A0[E56B0100]         <1>       mov    al, [audio_user]
                             <1>       ; 14/04/2017
000103C8 E8DB040000           <1>       call   set_irq_callback_service
                             <1>       ; 16/04/2017
000103CD A3[64030300]         <1>       mov    [u.r0], eax
                             <1>       ;jnc   sysret
000103D2 7316                 <1>       jnc    short sndc_init2 ; 21/04/2017
                             <1>       ;
000103D4 A3[C8030300]         <1>       mov    dword [u.error], eax
                             <1>
000103D9 A0[BF6B0100]         <1>       mov    al, [audio_intr] ; IRQ number
000103DE 31DB                 <1>       xor    ebx, ebx ; reset IRQ handler address
000103E0 E8F6FCFFFF           <1>       call   set_dev_IRQ_service
                             <1>
000103E5 E9D4C2FFFF           <1>       jmp    error
                             <1>
                             <1> sndc_init2:
                             <1>       ; 21/04/2017
000103EA 8B0D[D86B0100]       <1>       mov    ecx, [audio_buff_size] ; audio buffer size
000103F0 D1E1                 <1>       shl    ecx, 1 ; *2
000103F2 A1[DC6B0100]         <1>       mov    eax, [audio_dma_buff]
000103F7 21C0                 <1>       and    eax, eax
000103F9 7415                 <1>       jz     short sndc_init3
                             <1>
000103FB 8B15[E06B0100]       <1>       mov    edx, [audio_dmabuff_size] ; dma buffer size
00010401 39D1                 <1>       cmp    ecx, edx
00010403 744D                 <1>       je     short sndc_init5
                             <1>
00010405 87CA                 <1>       xchg   ecx, edx
00010407 E82552FFFF           <1>       call   deallocate_memory_block
0001040C 87D1                 <1>       xchg   edx, ecx
0001040E 31C0                 <1>       xor    eax, eax
                             <1> sndc_init3:
                             <1>       ; 12/05/2017
00010410 803D[BD6B0100]01     <1>       cmp    byte [audio_device], 1 ; SB 16
00010417 7515                 <1>       jne    short sndc_init4
00010419 C705[DC6B0100]-      <1>       mov    dword [audio_dma_buff], sb16_dma_buffer
0001041F [00000200]           <1>
00010423 C705[E06B0100]0000-  <1>       mov    dword [audio_dmabuff_size], 65536
0001042B 0100                 <1>
                             <1>       ;xor   eax, eax
                             <1>       ;mov   [u.r0], eax ; 0 = no error, successful
0001042D C3                   <1>       retn
                             <1>
                             <1> sndc_init4:
                             <1>       ; EAX = Beginning address (physical)
                             <1>       ; EAX = 0 -> Allocate mem block from the 1st proper aperture
                             <1>       ; ECX = Number of bytes to be allocated (>0)
0001042E E8F14FFFFF           <1>       call   allocate_memory_block
00010433 0F8296FDFFFF         <1>       jc     sound_buff_error
                             <1>
                             <1>       ; set dma buffer address and size parameters
00010439 A3[DC6B0100]         <1>       mov    [audio_dma_buff], eax ; dma buffer address
0001043E 890D[E06B0100]       <1>       mov    [audio_dmabuff_size], ecx ; dma buffer size
                             <1> ;     ; EAX = Beginning (physical) addr of the allocated mem block
                             <1> ;     ; ECX = Num of allocated bytes (rounded up to page borders)
                             <1> ;     cmp    byte [audio_pci], 0 ; AC97 audio controller ?
                             <1> ;     ja     short sndc_init4
                             <1> ;
                             <1> ;     ; Sound Blaster 16 uses classic DMA
                             <1> ;     mov    edx, eax
                             <1> ;     add    edx, ecx
                             <1> ;     cmp    edx, 1000000h ; 1st 16 MB
                             <1> ;     jna    short sndc_init4
                             <1> ;
                             <1> ;     ; error !
                             <1> ;     ; restore Memory Allocation Table Content
                             <1> ;     ; EAX = Beginning address (physical)
                             <1> ;     ; ECX = Number of bytes to be deallocated
                             <1> ;     call   deallocate_memory_block
                             <1> ;     ; reset dma buffer address and size parameters
                             <1> ;     xor    eax, eax ; 0
                             <1> ;     mov    [audio_dma_buff], eax ; 0
                             <1> ;     mov    [audio_dmabuff_size], ecx ; 0
                             <1> ;     jmp    sound_buff_error
                             <1> ;
```

```
                             <1> ;sndc_init4:
00010444 803D[BD6B0100]03    <1>        cmp    byte [audio_device], 3
                             <1>        ;jne   short sndc_init5
0001044B 7506                <1>        jne    short sndc_init14 ; 28/05/2017
0001044D E86A190000          <1>        call   set_vt8233_bdl
                             <1> sndc_init5:
                             <1>        ;sub   eax, eax ; 0
                             <1>        ;mov   [u.r0], eax ; 0 = no error, successful
00010452 C3                  <1>        retn
                             <1> sndc_init14:
00010453 E8311F0000          <1>        call   set_ac97_bdl
                             <1>        ;jmp   short sndc_init5
00010458 C3                  <1>        retn
                             <1>
                             <1> sound_play:
                             <1>        ; FUNCTION = 4
                             <1>        ; bl = Mode
                             <1>        ;       bit 0 = mono/stereo (1 = stereo)
                             <1>        ;       bit 1 = 8 bit / 16 bit (1 = 16 bit)
                             <1>        ; cx = Sampling Rate (Hz)
                             <1>
                             <1>        ; 13/06/2017
                             <1>        ; Note: Even if Mode bits are not 11b,
                             <1>        ;      AC'97 Audio Controller (&Codec)
                             <1>        ;      will play audio samples as 16 bit, stereo
                             <1>        ;      samples.
                             <1>        ;      (Program must fill the audio buffer
                             <1>        ;      as required; 8 bit samples must be converted
                             <1>        ;      to 16 bit samples and mono samples must be
                             <1>        ;      converted to stereo samples...)
                             <1>        ;
                             <1>        ; 28/05/2017
                             <1>        ; 15/05/2017, 20/05/2017
                             <1>        ; 21/04/2017, 24/04/2017
                             <1>        ; ... device check at first
00010459 A0[BD6B0100]        <1>        mov    al, [audio_device]
0001045E 08C0                <1>        or     al, al ; 0 ; pc speaker or invalid
00010460 0F843519FFFF        <1>        jz     beeper_gfx ; 'video.s' ; temporary !
                             <1> ;      cmp    al, 3 ; VIA VT 8237R (vt8233)
                             <1> ;      je     short snd_play_1
                             <1> ;      cmp    al, 1 ; SB 16
                             <1> ;      jne    soundc_dev_err ; temporary !
                             <1> ;snd_play_0:
                             <1>        ; ... buffer & (buffer) owner check at second
00010466 833D[D06B0100]00    <1>        cmp    dword [audio_buffer], 0
0001046D 0F865CFDFFFF        <1>        jna    sound_buff_error
00010473 A0[B3030300]        <1>        mov    al, [u.uno]
00010478 3A05[E56B0100]      <1>        cmp    al, [audio_user]
0001047E 0F858FFEFFFF        <1>        jne    sndc_owner_error
                             <1>
00010484 66890D[EE6B0100]    <1>        mov    [audio_freq], cx ; sample frequency (Hertz)
0001048B 88D8                <1>        mov    al, bl
0001048D 2401                <1>        and    al, 1 ; mono/stereo (1= stereo)
0001048F FEC0                <1>        inc    al ; channels
00010491 A2[ED6B0100]        <1>        mov    [audio_stmo], al ; sound channels (1 or 2)
00010496 B008                <1>        mov    al, 8
00010498 F6C302              <1>        test   bl, 2 ; bits per sample (1= 16 bit)
0001049B 7402                <1>        jz     short snd_play_bps
0001049D D0E0                <1>        shl    al, 1
                             <1> snd_play_bps:
0001049F A2[EC6B0100]        <1>        mov    [audio_bps], al
                             <1>        ; Transfer ring 3 (user's) audio buffer content to dma buffer
000104A4 8B3D[DC6B0100]      <1>        mov    edi, [audio_dma_buff] ; dma buffer (ring 0)
000104AA 09FF                <1>        or     edi, edi
000104AC 0F841DFDFFFF        <1>        jz     sound_buff_error
000104B2 8B35[D46B0100]      <1>        mov    esi, [audio_p_buffer] ; physical address (ring 3)
000104B8 8B0D[D86B0100]      <1>        mov    ecx, [audio_buff_size] ; 15/05/2017
                             <1>        ;rep   movsb
000104BE C1E902              <1>        shr    ecx, 2
000104C1 F3A5                <1>        rep    movsd
                             <1>        ; 20/05/2017
000104C3 C605[E46B0100]01    <1>        mov    byte [audio_flag], 1 ; next half (on next time)
                             <1>
                             <1>        ; 24/04/2017
000104CA A0[BD6B0100]        <1>        mov    al, [audio_device]
000104CF 3C03                <1>        cmp    al, 3 ; VT8233 (VT8237R)
000104D1 7410                <1>        je     short snd_play_1
000104D3 3C01                <1>        cmp    al, 1 ; Sound Blaster 16
000104D5 7512                <1>        jne    short snd_play_2  ; 28/05/2017
000104D7 E8E81A0000          <1>        call   SbInit_play
000104DC 0F82F4FCFFFF        <1>        jc     soundc_respond_err
000104E2 C3                  <1>        retn
                             <1>
                             <1> snd_play_1:
000104E3 E804190000          <1>        call   vt8233_start_play
000104E8 C3                  <1>        retn
                             <1>
                             <1> snd_play_2:
                             <1>        ; 28/05/2017
                             <1>        ;cmp   al, 2 ; AC'97
                             <1>        ;jne   short snd_play_3
                             <1>
000104E9 E8CF1E0000          <1>        call   ac97_start_play
000104EE C3                  <1>        retn
                             <1>
                             <1> ;snd_play_3:
                             <1> ;      ;call  hda_start_play
                             <1> ;      retn
                             <1>
                             <1> sound_pause:
                             <1>        ; FUNCTION = 5
                             <1>        ; Pause
                             <1>        ; 28/05/2017
```

```
                              <1>        ; 24/04/2017
                              <1>        ; 22/04/2017
000104EF E814030000           <1>        call   snd_dev_check
000104F4 7275                 <1>        jc     short snd_nothing ; temporary.
000104F6 E81A030000           <1>        call   snd_buf_check
000104FB 726E                 <1>        jc     short snd_nothing ; temporary.
000104FD A0[BD6B0100]         <1>        mov    al, [audio_device]
00010502 3C03                 <1>        cmp    al, 3 ; VIA VT 8237R (vt8233)
00010504 7409                 <1>        je     short snd_pause_1
00010506 3C01                 <1>        cmp    al, 1 ; Sound Blaster 16
00010508 750A                 <1>        jne    short snd_pause_2 ; 28/05/2017
0001050A E9931C0000           <1>        jmp    sb16_pause
                              <1> snd_pause_1:
0001050F E996190000           <1>        jmp    vt8233_pause
                              <1> snd_pause_2:
                              <1>        ; 28/05/2017
                              <1>        ;cmp   al, 2 ; AC'97
                              <1>        ;jne   short snd_nothing ; temporary.
00010514 E932200000           <1>        jmp    ac97_pause
                              <1>
                              <1> sound_continue:
                              <1>        ; FUNCTION = 6
                              <1>        ; Continue to play
                              <1>        ; 28/05/2017
                              <1>        ; 22/04/2017
00010519 E8EA020000           <1>        call   snd_dev_check
0001051E 724B                 <1>        jc     short snd_nothing ; temporary.
00010520 E8F0020000           <1>        call   snd_buf_check
00010525 7244                 <1>        jc     short snd_nothing ; temporary.
00010527 A0[BD6B0100]         <1>        mov    al, [audio_device]
0001052C 3C03                 <1>        cmp    al, 3 ; VIA VT 8237R (vt8233)
0001052E 7409                 <1>        je     short snd_cont_1
00010530 3C01                 <1>        cmp    al, 1 ; Sound Blaster 16
00010532 750A                 <1>        jne    short snd_cont_2 ; 28/05/2017
00010534 E98C1C0000           <1>        jmp    sb16_continue
                              <1> snd_cont_1:
00010539 E919190000           <1>        jmp    vt8233_play
                              <1> snd_cont_2:
                              <1>        ; 28/05/2017
                              <1>        ;cmp   al, 2 ; AC'97
                              <1>        ;jne   short snd_nothing ; temporary.
0001053E E9D01E0000           <1>        jmp    ac97_play
                              <1>
                              <1> sound_stop:
                              <1>        ; FUNCTION = 7
                              <1>        ; Stop playing
                              <1>        ; 28/05/2017
                              <1>        ; 24/05/2017
                              <1>        ; 21/04/2017, 22/04/2017, 24/04/2017
00010543 E8C0020000           <1>        call   snd_dev_check
00010548 7221                 <1>        jc     short snd_nothing ; temporary.
                              <1>        ;call  snd_buf_check
0001054A E8CF020000           <1>        call   snd_user_check ; 24/05/2017
0001054F 721A                 <1>        jc     short snd_nothing ; temporary.
                              <1>
00010551 A0[BD6B0100]         <1>        mov    al, [audio_device]
00010556 3C03                 <1>        cmp    al, 3 ; VIA VT 8237R (vt8233)
00010558 0F8455180000         <1>        je     vt8233_stop
                              <1>        ; 28/05/2017
                              <1>        ;ja    short snd_nothing
0001055E 3C01                 <1>        cmp    al, 1 ; Sound Blaster 16
00010560 0F84821C0000         <1>        je     sb16_stop
                              <1>        ;cmp   al, 2
                              <1>        ;je    short ac97_stop
00010566 E9B21F0000           <1>        jmp    ac97_stop ; temporary.
                              <1>        ;jmp   hda_stop
                              <1>
                              <1> snd_nothing:
                              <1>        ; 21/04/2017
0001056B C3                   <1>        retn
                              <1>
                              <1> soundc_reset:
                              <1>        ; FUNCTION = 8
                              <1>        ; Reset Audio Controller
                              <1>        ; 28/05/2017
                              <1>        ; 22/04/2017
0001056C E897020000           <1>        call   snd_dev_check
00010571 72F8                 <1>        jc     snd_nothing ; temporary.
00010573 E89D020000           <1>        call   snd_buf_check
00010578 72F1                 <1>        jc     snd_nothing ; temporary.
                              <1>
0001057A A0[BD6B0100]         <1>        mov    al, [audio_device]
0001057F 3C03                 <1>        cmp    al, 3 ; VIA VT 8237R (vt8233)
00010581 0F8431190000         <1>        je     vt8233_reset
00010587 77E2                 <1>        ja     short snd_nothing ; temporary.
                              <1>        ;ja    hda_reset
00010589 3C01                 <1>        cmp    al, 1 ; Sound Blaster 16
0001058B 0F850B200000         <1>        jne    ac97_reset
00010591 E8A41C0000           <1>        call   sb16_reset
00010596 0F823AFCFFFF         <1>        jc     soundc_respond_err
0001059C C3                   <1>        retn
                              <1>
                              <1> soundc_cancel:
                              <1>        ; FUNCTION = 9
                              <1>        ; Cancel audio callback service
                              <1>        ; 22/04/2017
0001059D A0[E56B0100]         <1>        mov    al, [audio_user]
000105A2 3A05[B3030300]       <1>        cmp    al, [u.uno]
000105A8 75C1                 <1>        jne    short snd_nothing
                              <1>        ; RESET (audio) INTERRUPT CALLBACK SERVICE
000105AA 8A1D[BF6B0100]       <1>        mov    bl, [audio_intr] ; IRQ number
000105B0 A0[B3030300]         <1>        mov    al, [u.uno]
000105B5 28FF                 <1>        sub    bh, bh ; 0 ; unlink IRQ from user service
```

```
000105B7 E8EC020000           <1>       call   set_irq_callback_service
000105BC 0F8256FDFFFF         <1>       jc     sndc_perm_error ; 'permission denied' error
000105C2 C3                   <1>       retn
                              <1>
                              <1> sound_dalloc:
                              <1>       ; FUNCTION = 10
                              <1>       ; Deallocate (ring 3) audio buffer
                              <1>       ; 22/04/2017
000105C3 A0[E56B0100]         <1>       mov    al, [audio_user]
000105C8 3A05[B3030300]       <1>       cmp    al, [u.uno]
000105CE 759B                 <1>       jne    short snd_nothing
000105D0 8B1D[D06B0100]       <1>       mov    ebx, [audio_buffer]
                              <1>       ;or    ebx, ebx
                              <1>       ;jz    short snd_nothing
000105D6 8B0D[D86B0100]       <1>       mov    ecx, [audio_buff_size]
000105DC E89E51FFFF           <1>       call   deallocate_user_pages
000105E1 31C0                 <1>       xor    eax, eax
000105E3 A3[D06B0100]         <1>       mov    [audio_buffer], eax ; 0
000105E8 A2[E56B0100]         <1>       mov    [audio_user], al ; 0
000105ED C3                   <1>       retn
                              <1>
                              <1> sound_volume:
                              <1>       ; FUNCTION = 11
                              <1>       ; Set sound volume level
                              <1>       ; 28/05/2017
                              <1>       ; 20/05/2017
                              <1>       ; 22/04/2017, 24/04/2017
                              <1>       ; bl = component (0 = master/playback/lineout volume)
                              <1>       ; cl = left channel volume level (0 to 31)
                              <1>       ; ch = right channel volume level (0 to 31)
                              <1>
000105EE 80FB80               <1>       cmp    bl, 80h
000105F1 720E                 <1>       jb     short snd_vol_1
000105F3 0F8772FFFFFF         <1>       ja     snd_nothing ; temporary.
                              <1>       ; Set volume level for next play (BL>= 80h)
000105F9 66890D[F26B0100]     <1>       mov    [audio_master_volume], cx
00010600 C3                   <1>       retn
                              <1> snd_vol_1:
                              <1>       ; set volume level immediate (BL< 80h)
00010601 80FB00               <1>       cmp    bl, 0
00010604 0F8761FFFFFF         <1>       ja     snd_nothing ; temporary.
                              <1>
0001060A E8F9010000           <1>       call   snd_dev_check
0001060F 0F8256FFFFFF         <1>       jc     snd_nothing ; temporary.
00010615 E8FB010000           <1>       call   snd_buf_check
0001061A 0F824BFFFFFF         <1>       jc     snd_nothing ; temporary.
                              <1>
00010620 A0[BD6B0100]         <1>       mov    al, [audio_device]
00010625 3C03                 <1>       cmp    al, 3 ; VIA VT 8237R (vt8233)
00010627 0F84A4180000         <1>       je     vt8233_volume
                              <1>       ; 28/05/2017
0001062D 0F8738FFFFFF         <1>       ja     snd_nothing ; temporary.
                              <1>       ;ja    hda_volume
                              <1>       ; Sound Blaster 16
00010633 3C01                 <1>       cmp    al, 1 ; SB 16
00010635 0F84321B0000         <1>       je     sb16_volume
0001063B E9EF1D0000           <1>       jmp    ac97_volume
                              <1>
                              <1> soundc_disable:
                              <1>       ; FUNCTION = 12
                              <1>       ; Disable audio device (and unlink DMA memory)
                              <1>       ; 28/05/2017
                              <1>       ; 24/05/2017
                              <1>       ; 22/04/2017
00010640 E8C3010000           <1>       call   snd_dev_check
00010645 0F827DFBFFFF         <1>       jc     soundc_dev_err ; temporary.
                              <1>       ;call  snd_buf_check
                              <1>       ;jc    sndc_owner_error ; temporary.
                              <1>
0001064B A0[BD6B0100]         <1>       mov    al, [audio_device]
00010650 3C03                 <1>       cmp    al, 3 ; VIA VT 8237R (vt8233)
00010652 7418                 <1>       je     short snd_disable_1
00010654 0F8711FFFFFF         <1>       ja     snd_nothing ; temporary.
0001065A 3C01                 <1>       cmp    al, 1 ; Sound Blaster 16
0001065C 7507                 <1>       jne    short snd_disable_0
0001065E E8851B0000           <1>       call   sb16_stop
00010663 EB0C                 <1>       jmp    short snd_disable_2
                              <1> snd_disable_0:
00010665 E8B31E0000           <1>       call   ac97_stop
0001066A EB05                 <1>       jmp    short snd_disable_2
                              <1> snd_disable_1:
0001066C E842170000           <1>       call   vt8233_stop
                              <1> snd_disable_2:
00010671 A0[BF6B0100]         <1>       mov    al, [audio_intr]
00010676 29DB                 <1>       sub    ebx, ebx ; 0 = reset
00010678 E85EFAFFFF           <1>       call   set_dev_IRQ_service
                              <1>
                              <1>       ;mov   al, [audio_intr]
0001067D 28E4                 <1>       sub    ah, ah ; 0 = reset
0001067F E8C0F6FFFF           <1>       call   set_hardware_int_vector
                              <1>
00010684 31C0                 <1>       xor    eax, eax
00010686 A2[BD6B0100]         <1>       mov    byte [audio_device], al
0001068B A2[BF6B0100]         <1>       mov    byte [audio_intr], al
00010690 8705[DC6B0100]       <1>       xchg   eax, [audio_dma_buff]
                              <1>       ; 24/05/2017
                              <1>       ;or    eax, eax
                              <1>       ;jz    short snd_disable_3
                              <1>       ;cmp   eax, sb16_dma_buffer ; default DMA buffer
                              <1>       ;je    short snd_disable_3
00010696 803D[BC6B0100]00     <1>       cmp    byte [audio_pci], 0 ; AC97 audio controller ?
0001069D 7612                 <1>       jna    short snd_disable_3
0001069F C605[BC6B0100]00     <1>       mov    byte [audio_pci], 0
```

```
                                 <1>       ;sub   ecx, ecx
                                 <1>       ;xchg  ecx, [audio_dmabuff_size]
000106A6 8B0D[E06B0100]          <1>       mov    ecx, [audio_dmabuff_size]
000106AC E8804FFFFF              <1>       call   deallocate_memory_block
                                 <1> snd_disable_3:
000106B1 C3                      <1>       retn
                                 <1>
                                 <1> sound_dma_map:
                                 <1>       ; FUNCTION = 13
                                 <1>       ; Map audio dma buff addr to user's buffer addr
                                 <1>       ; 12/05/2017
000106B2 21C9                    <1>       and    ecx, ecx
000106B4 0F8415FBFFFF            <1>       jz     sound_buff_error
000106BA 803D[BD6B0100]01        <1>       cmp    byte [audio_device], 1
000106C1 7229                    <1>       jb     short snd_dma_map_1
                                 <1> snd_dma_map_0:
000106C3 A1[DC6B0100]            <1>       mov    eax, [audio_dma_buff]
000106C8 21C0                    <1>       and    eax, eax
000106CA 7420                    <1>       jz     short snd_dma_map_1
                                 <1>       ;
000106CC 8A1D[E56B0100]          <1>       mov    bl, [audio_user]
000106D2 08DB                    <1>       or     bl, bl
000106D4 7416                    <1>       jz     short snd_dma_map_1
000106D6 3A1D[B3030300]          <1>       cmp    bl, [u.uno]
000106DC 0F8531FCFFFF            <1>       jne    sndc_owner_error
                                 <1>       ;
000106E2 8B1D[E06B0100]          <1>       mov    ebx, [audio_dmabuff_size]
000106E8 21DB                    <1>       and    ebx, ebx
000106EA 750A                    <1>       jnz    short snd_dma_map_2
                                 <1> snd_dma_map_1:
000106EC B8[00000200]            <1>       mov    eax, sb16_dma_buffer
000106F1 BB00000100              <1>       mov    ebx, 65536
                                 <1> snd_dma_map_2:
000106F6 81C1FF0F0000            <1>       add    ecx, PAGE_SIZE-1 ; 4095
000106FC 6681E100F0              <1>       and    cx, ~PAGE_OFF ; not 4095
00010701 39D9                    <1>       cmp    ecx, ebx
00010703 0F87C6FAFFFF            <1>       ja     sound_buff_error
00010709 50                      <1>       push   eax
0001070A 89D3                    <1>       mov    ebx, edx
0001070C C1E90C                  <1>       shr    ecx, 12 ; byte count to page count
                                 <1>       ; eax = physical address of (audio) dma buffer
                                 <1>       ; ebx = virtual address of (audio) dma buffer (user's pgdir)
                                 <1>       ; ecx = page count (>0)
0001070F E88D4FFFFF              <1>       call   direct_memory_access
00010714 58                      <1>       pop    eax
00010715 0F82B4FAFFFF            <1>       jc     sound_buff_error
0001071B A3[64030300]            <1>       mov    [u.r0], eax
00010720 C3                      <1>       retn
                                 <1>
                                 <1> soundc_info:
                                 <1>       ; FUNCTION = 14
                                 <1>       ; Get Audio Controller Info
                                 <1>       ; 10/06/2017
                                 <1>       ; 05/06/2017
00010721 20DB                    <1>       and    bl, bl ; 0
00010723 740A                    <1>       jz     short sndc_info_0
                                 <1>       ; invalid parameter !
00010725 B817000000              <1>       mov    eax, ERR_INV_PARAMETER ; 23
                                 <1> ;sndc_inf_error:
                                 <1> ;     mov    [u.r0], eax
                                 <1> ;     mov    [u.error], eax
                                 <1> ;     jmp    error
0001072A E9ACFAFFFF              <1>       jmp    sysaudio_err
                                 <1>
                                 <1> sndc_info_0:
0001072F E8D4000000              <1>       call   snd_dev_check
00010734 0F828EFAFFFF            <1>       jc     soundc_dev_err
                                 <1>
0001073A 8B1D[C86B0100]          <1>       mov    ebx, [audio_vendor]
00010740 8B0D[C46B0100]          <1>       mov    ecx, [audio_dev_id]
                                 <1>       ;mov   al,  [audio_device]
00010746 3C02                    <1>       cmp    al, 2 ; AC'97 (ICH)
00010748 7513                    <1>       jne    short sndc_info_1
                                 <1>       ; Intel AC97 (ICH) Audio Controller (=2)
0001074A 668B15[F66B0100]        <1>       mov    dx, [NABMBAR]
00010751 C1E210                  <1>       shl    edx, 16
00010754 668B15[F46B0100]        <1>       mov    dx, [NAMBAR]
0001075B EB07                    <1>       jmp    short sndc_info_2
                                 <1> sndc_info_1:
                                 <1>       ; 05/06/2017
                                 <1>       ; Note: Intel HDA code (here) is not ready yet!
                                 <1>       ; !!! SB16 or VT8233 (VT8237R) !!!
0001075D 0FB715[C26B0100]        <1>       movzx  edx, word [audio_io_base]
                                 <1> sndc_info_2:
00010764 88C4                    <1>       mov    ah, al ;  [audio_device]
00010766 A0[BF6B0100]            <1>       mov    al, [audio_intr]
                                 <1>
                                 <1>       ; EAX = IRQ Number in AL
                                 <1>       ;      Audio Device Number in AH
                                 <1>       ; EBX = DEV/VENDOR ID
                                 <1>       ;      (DDDDDDDDDDDDDDDDVVVVVVVVVVVVVVVV)
                                 <1>       ; ECX = BUS/DEV/FN
                                 <1>       ;      (00000000BBBBBBBBDDDDDFFF00000000)
                                 <1>       ; EDX = NABMBAR/NAMBAR (for AC97)
                                 <1>       ;      (Low word, DX = NAMBAR address)
                                 <1>       ; EDX = Base IO Addr (DX) for SB16 & VT8233
                                 <1>
                                 <1>       ; 10/06/2017
0001076B A3[64030300]            <1>       mov    [u.r0], eax
00010770 8B2D[60030300]          <1>       mov    ebp, [u.usp]
00010776 895D10                  <1>       mov    [ebp+16], ebx  ; ebx
00010779 895514                  <1>       mov    [ebp+20], edx  ; edx
0001077C 894D18                  <1>       mov    [ebp+24], ecx  ; ecx
```

```
                             <1>
0001077F C3                  <1>       retn
                             <1>
                             <1> sound_data:
                             <1>       ; FUNCTION = 15
                             <1>       ; Get Current Sound data for graphics
                             <1>       ; 22/06/2017
                             <1>       ;
00010780 E883000000          <1>       call   snd_dev_check
00010785 0F823DFAFFFF        <1>       jc     soundc_dev_err ; Device not ready !
                             <1>
0001078B 80FB00              <1>       cmp    bl, 0
0001078E 760A                <1>       jna    short sound_data_0
                             <1>
                             <1>       ; Only PCM OUT buffer data is valid for now!
00010790 B817000000          <1>       mov    eax, ERR_INV_PARAMETER  ; 23
00010795 E941FAFFFF          <1>       jmp    sysaudio_err
                             <1>
                             <1> sound_data_0:
0001079A A1[DC6B0100]        <1>       mov    eax, [audio_dma_buff]
0001079F 09C0                <1>       or     eax, eax
000107A1 0F8428FAFFFF        <1>       jz     sound_buff_error
                             <1>
000107A7 803D[BD6B0100]04    <1>       cmp    byte [audio_device], 4 ; Intel HDA
000107AE 744F                <1>       je     short sound_data_4 ; temporary ! (22/06/2017)
                             <1>
000107B0 21C9                <1>       and    ecx, ecx
                             <1>       ;jnz   short sound_data_1 ; sample tranfer
                             <1>
                             <1>       ; Return only DMA Buffer pointer/offset...
                             <1>       ; (If DMA Buffer has been mapped to user's
                             <1>       ;  memory space; program can get graphics
                             <1>       ;  data by using only this pointer value.)
                             <1>
                             <1>       ;call  get_dma_buffer_offset
                             <1>       ;; eax = DMA buffer offset
                             <1>       ;;     (!not half buffer offset!)
                             <1>       ;mov   [u.r0], eax
                             <1>       ;retn
                             <1>
000107B2 0F845C1F0000        <1>       jz     get_dma_buffer_offset
                             <1>
                             <1> sound_data_1:
                             <1>       ;mov   eax, [audio_dmabuff_size]
                             <1>       ;shr   eax, 1 ; half buffer size
                             <1>       ;cmp   ecx, eax
                             <1>       ;ja    short sound_buff_error
                             <1>
000107B8 3B0D[E06B0100]      <1>       cmp    ecx, [audio_dmabuff_size]
000107BE 0F870BFAFFFF        <1>       ja     sound_buff_error
                             <1>
000107C4 89D0                <1>       mov    eax, edx
000107C6 25FF0F0000          <1>       and    eax, PAGE_OFF ; 4095 (0FFFh)
000107CB 81F900100000        <1>       cmp    ecx, 4096
000107D1 7605                <1>       jna    short sound_data_2
000107D3 B900100000          <1>       mov    ecx, 4096 ; max. 1 page
                             <1> sound_data_2:
000107D8 01C8                <1>       add    eax, ecx
000107DA 3D00100000          <1>       cmp    eax, 4096
000107DF 7606                <1>       jna    short sound_data_3
000107E1 6625FF0F            <1>       and    ax, PAGE_OFF ; 4095 (0FFFh)
000107E5 29C1                <1>       sub    ecx, eax
                             <1>       ; here, ECX has been adjusted to fit
                             <1>       ;      in page border..  (<= 4096, >0)
                             <1> sound_data_3:
000107E7 51                  <1>       push   ecx
000107E8 52                  <1>       push   edx
000107E9 89D3                <1>       mov    ebx, edx
000107EB E89F4AFFFF          <1>       call   get_physical_addr
000107F0 5A                  <1>       pop    edx
000107F1 59                  <1>       pop    ecx
000107F2 0F82D7F9FFFF        <1>       jc     sound_buff_error
                             <1>
                             <1>       ; eax = physical address of user's buffer
000107F8 89C3                <1>       mov    ebx, eax
                             <1>       ; ecx = byte (transfer) count
                             <1>       ;call  get_current_sound_data
                             <1>       ;retn
000107FA E9721E0000          <1>       jmp    get_current_sound_data
                             <1>
                             <1> sound_data_4:
                             <1>       ; Intel HDA code is not ready yet !
                             <1>       ; 22/06/2017
000107FF 31C0                <1>       xor    eax, eax
00010801 48                  <1>       dec    eax
00010802 A3[64030300]        <1>       mov    [u.r0], eax ; 0FFFFFFFFh
00010807 C3                  <1>       retn
                             <1>
                             <1> snd_dev_check:
                             <1>       ; 10/06/2017
                             <1>       ; 05/06/2017
                             <1>       ; 24/05/2017
                             <1>       ; 22/04/2017
                             <1>       ; 21/04/2017
                             <1>       ; ... device check at first
00010808 A0[BD6B0100]        <1>       mov    al, [audio_device]
0001080D 3C01                <1>       cmp    al, 1 ; SB 16
0001080F 7203                <1>       jb     short snd_dev_chk_retn ; error !
                             <1>       ;cmp   al, 4 ; Intel HDA
                             <1>       ;ja    short snd_dbchk_stc ; invalid !
                             <1>       ; 10/06/2017
00010811 3C05                <1>       cmp    al, 5
00010813 F5                  <1>       cmc
```

```
                             <1> snd_dev_chk_retn:
00010814 C3                  <1>        retn
                             <1>
                             <1> snd_buf_check:
                             <1>        ; 10/06/2017
                             <1>        ; 22/04/2017
                             <1>        ; 21/04/2017
                             <1>        ; ... buffer & (buffer) owner check at second
00010815 833D[D06B0100]00    <1>        cmp    dword [audio_buffer], 0
0001081C 760D                <1>        jna    short snd_dbchk_stc
                             <1> snd_user_check:
0001081E A0[B3030300]        <1>        mov    al, [u.uno]
00010823 3A05[E56B0100]      <1>        cmp    al, [audio_user]
                             <1>        ;jne   short snd_dbchk_stc
                             <1>        ;retn
00010829 74E9                <1>        je     short snd_dev_chk_retn
                             <1>
                             <1> snd_dbchk_stc:
0001082B F9                  <1>        stc
0001082C C3                  <1>        retn
                             <1>
                             <1> sound_update:
                             <1>        ; FUNCTION = 16
                             <1>        ; bl =
                             <1>        ;    0 = automatic (sequental) update (with flag switch!)
                             <1>        ;    1 = update dma half buffer 1 (without flag switch!)
                             <1>        ;    2 = update dma half buffer 2 (without flag switch!)
                             <1>        ;  FFh = get current flag value
                             <1>        ;      0 = dma half buffer 1 (will be played next)
                             <1>        ;      1 = dma half buffer 2 (will be played next)
                             <1>
                             <1>        ; 10/10/2017
                             <1>
                             <1>        ; ... device check at first
0001082D A0[BD6B0100]        <1>        mov    al, [audio_device]
00010832 08C0                <1>        or     al, al ; 0 ; pc speaker or invalid
00010834 0F848EF9FFFF        <1>        jz     soundc_dev_err
                             <1>
                             <1>        ; ... buffer & (buffer) owner check at second
0001083A 833D[D06B0100]00    <1>        cmp    dword [audio_buffer], 0
00010841 0F8688F9FFFF        <1>        jna    sound_buff_error
00010847 A0[B3030300]        <1>        mov    al, [u.uno]
0001084C 3A05[E56B0100]      <1>        cmp    al, [audio_user]
00010852 0F85BBFAFFFF        <1>        jne    sndc_owner_error
                             <1>
                             <1>        ; Transfer ring 3 (user's) audio buffer content to dma buffer
00010858 8B3D[DC6B0100]      <1>        mov    edi, [audio_dma_buff] ; dma buffer (ring 0)
0001085E 09FF                <1>        or     edi, edi
00010860 0F8469F9FFFF        <1>        jz     sound_buff_error
00010866 8B35[D46B0100]      <1>        mov    esi, [audio_p_buffer] ; physical address (ring 3)
0001086C 8B0D[D86B0100]      <1>        mov    ecx, [audio_buff_size]
                             <1>
                             <1>        ;movzx eax, byte [audio_flag]
00010872 A0[E46B0100]        <1>        mov    al, [audio_flag]
                             <1>
00010877 FEC3                <1>        inc    bl
00010879 7427                <1>        jz     short snd_update_3 ; bl = 0FFh
0001087B FECB                <1>        dec    bl
0001087D 7411                <1>        jz     short snd_update_0 ; bl = 0
                             <1>
0001087F 80FB02              <1>        cmp    bl, 2
00010882 7417                <1>        je     short snd_update_1 ; dma half buffer 2
00010884 7217                <1>        jb     short snd_update_2 ; dma half buffer 1
                             <1>
                             <1>        ; invalid parameter !
00010886 B817000000          <1>        mov    eax, ERR_INV_PARAMETER ; 23
                             <1> ;      mov    [u.r0], eax
                             <1> ;      mov    [u.error], eax
                             <1> ;      jmp    error
0001088B E94BF9FFFF          <1>        jmp    sysaudio_err
                             <1>
                             <1> snd_update_0:
00010890 8035[E46B0100]01    <1>        xor    byte [audio_flag], 1 ; update flag !!!
00010897 3C01                <1>        cmp    al, 1
00010899 7202                <1>        jb     short snd_update_2 ; dma half buffer 1
                             <1> snd_update_1:
                             <1>        ; dma half buffer 2
0001089B 01CF                <1>        add    edi, ecx
                             <1> snd_update_2:
                             <1>        ;rep   movsb
0001089D C1E902              <1>        shr    ecx, 2
000108A0 F3A5                <1>        rep    movsd
                             <1> snd_update_3:
000108A2 A3[64030300]        <1>        mov    [u.r0], eax
                             <1>
000108A7 C3                  <1>        retn
                             <1>
                             <1>
                             <1> set_irq_callback_service:
                             <1>        ; 10/06/2017
                             <1>        ; 12/05/2017
                             <1>        ; 24/04/2017
                             <1>        ; 22/04/2017
                             <1>        ; caller: 'syscalbac' or 'sysaudio' or ...
                             <1>        ; 13/04/2017, 14/04/2017, 17/04/2017
                             <1>        ; 24/02/2017, 26/02/2017, 28/02/2017
                             <1>        ; 21/02/2017 - TRDOS 386 (TRDOS v2.0)
                             <1>        ;
                             <1>        ; Link or unlink IRQ callback service to/from user (ring 3)
                             <1>        ;
                             <1>        ; INPUT ->
                             <1>        ;      If AL = 0, the caller is 'syscalbac';
                             <1>        ;         otherwise, the caller is 'sysaudio' or ...
                             <1>        ;         (AL = user number)
                             <1>        ;
                             <1>        ;      BL = IRQ number (Hardware interrupt request number)
                             <1>        ;           (0 t0 15 but IRQ 0,1,2,6,8,14,15 are prohibited)
                             <1>        ;           IRQ numbers 3,4,5,7,9,10,11,12,13 are valid
                             <1>        ;           (numbers >15 are invalid)
                             <1>        ;
                             <1>        ;      BH = 0 = Unlink IRQ (in BL) from user (ring 3) service
                             <1>        ;           1 = Link IRQ by using Signal Response Byte method
                             <1>        ;           2 = Link IRQ by using Callback service method
                             <1>        ;           3 = Link IRQ by using Auto Increment S.R.B. method
                             <1>        ;          >3 = invalid
                             <1>        ;              (syscallback version will return to user)
                             <1>        ;
                             <1>        ;      CL = Signal Return/Response Byte value
                             <1>        ;
                             <1>        ;      If BH = 2, kernel will put a counter value
                             <1>        ;                (into the S.R.B. addr)
                             <1>        ;              between 0 to 255. (start value = CL+1)
                             <1>        ;
                             <1>        ;      NOTE: counter value, for example: even and odd numbers
                             <1>        ;            may be used for -audio- DMA buffer switch
                             <1>        ;            within double buffer method, etc.
                             <1>        ;
                             <1>        ;      EDX = Signal return (Response) byte address
                             <1>        ;                            - or -
                             <1>        ;            Interrupt/Callback service/routine address
                             <1>        ;
                             <1>        ;            (virtual address in user's memory space)
                             <1>        ;
                             <1>        ; OUTPUT ->
                             <1>        ;      CF = 0 & EAX = 0 -> Successful setting
                             <1>        ;      CF = 1 & EAX > 0 -> IRQ is prohibited or locked
                             <1>        ;                   by another process
                             <1>        ;              eax = ERR_PERM_DENIED -> prohibited or locked
                             <1>        ;              eax = ERR_INV_PARAMETER ->
                             <1>        ;                    invalid parameter/option or bad address
                             <1>        ;
                             <1>        ; TRDOS 386 - IRQ CALLBACK structures (parameters):
                             <1>        ;
                             <1>        ;         [u.irqlock] = 1 word, IRQ flags (0-15) that indicates
                             <1>        ;                   which IRQs are locked by (that) user.
                             <1>        ;                    Lock and unlock (by user) will change
                             <1>        ;                   these flags or 'terminate process' (sysexit)
                             <1>        ;                   will clear these flags and unlock those IRQs.
                             <1>        ;
                             <1>        ;                   Bit 0 is for IRQ 0 and Bit 15 is for IRQ 15
                             <1>        ;
                             <1>        ;         IRQ(x).owner      : 1 byte, user, [u.uno], 0 = free (unlocked)
                             <1>        ;
                             <1>        ;         IRQ(x).method : 1 byte for callback method & status
                             <1>        ;                      0 = Signal Response Byte method
                             <1>        ;                      1 = Callback service method
                             <1>        ;                      >1 = invalid for current 'syscalback'.
                             <1>        ;                   or(+) 80h = IRQ is in use by system (ring 0)
                             <1>        ;                               function (audio etc.) or
                             <1>        ;                              a device driver.
                             <1>        ;                   (system function will ignore the lock/owner)
                             <1>        ;
                             <1>        ;         IRQ(x).srb: 1 byte, Signal Return/Response byte value
                             <1>        ;                     (a fixed value by user or a counter value
                             <1>        ;                    from 0 to 255, which is increased by every
                             <1>        ;                    interrupt just before putting it into
                             <1>        ;                    the Signal Response byte address
                             <1>        ;                    (This is not used in callback serv method)
                             <1>        ;
                             <1>        ;         IRQ(x).addr      : 1 dword
                             <1>        ;                     Signal Response Byte address (physical)
                             <1>        ;                                -or-
                             <1>        ;                     Callback service address (virtual)
                             <1>        ;
                             <1>        ;         IRQ(x).dev: 1 byte
                             <1>        ;                     0 = Default device or kernel function
                             <1>        ;                                -or-
                             <1>        ;                     1-255 = Assigned device driver number
                             <1>        ;
                             <1>        ;         (x) = 3,4,5,7,9,10,11,12,13
                             <1>        ;
                             <1>
000108A8 80FB0F              <1>        cmp    bl, 15
000108AB 7729                <1>        ja     short scbs_2
                             <1>
000108AD 80FF03              <1>        cmp    bh, 3
000108B0 7724                <1>        ja     short scbs_2  ; invalid parameter
                             <1>
000108B2 0FB6FB              <1>        movzx  edi, bl ; save IRQ number
                             <1>
                             <1>               ;  IRQ 0,1,2,6,8,14,15 are prohibited
                             <1>        ;IRQenum: ; 'trdosk9.s'
                             <1>        ;      db  0,0,0,1,2,3,0,4,0,5,6,7,8,9,0,0
                             <1>
000108B5 0FB6B7[08160100]    <1>        movzx  esi, byte [edi+IRQenum] ; IRQ availability
                             <1>                                  ; enumeration/index
                             <1>        ;dec   esi
000108BC 664E                <1>        dec    si
000108BE 780F                <1>        js     short scbs_1 ;  0 -> 0FFFFh
                             <1>
                             <1>        ; ESI = IRQ callback parameters index number (0 to 8)
                             <1>
000108C0 08FF                <1>        or     bh, bh
000108C2 7419                <1>        jz     short scbs_4 ; unlink the IRQ (in BL)
                             <1>
000108C4 FECF                <1>        dec    bh
```

```
                             <1>        ; bh = method (0 = signal response byte, 1 = callback)
                             <1>        ;            (2 = auto increment of signal response byte)
                             <1>
000108C6 80BE[6E6B0100]00    <1>        cmp    byte [esi+IRQ.owner], 0 ; locked ?
000108CD 7637                <1>        jna    short scbs_6 ; no... OK...
                             <1>
                             <1> scbs_1:
                             <1>        ; permission denied (prohibited IRQ)
000108CF B80B000000          <1>        mov    eax, ERR_PERM_DENIED
000108D4 F9                  <1>        stc
000108D5 C3                  <1>        retn
                             <1> scbs_2:
000108D6 F9                  <1>        stc
                             <1> scbs_3:
000108D7 B817000000          <1>        mov    eax, ERR_INV_PARAMETER
000108DC C3                  <1>        retn
                             <1>
                             <1> scbs_4: ; unlink the requested IRQ (if it belongs to current user)
                             <1>        ; 10/06/2017
                             <1>        ; 22/04/2017
                             <1>        ; 14/04/2017
                             <1>        ; If AL = 0 -> The caller is 'syscalbac'
000108DD 8AA6[6E6B0100]      <1>        mov    ah, [esi+IRQ.owner]
000108E3 3A25[B3030300]      <1>        cmp    ah, [u.uno]
000108E9 75E4                <1>        jne    short scbs_1
                             <1>
000108EB FE0D[D6030300]      <1>        dec    byte [u.irqc] ; decrease IRQ count (in use)
                             <1>
                             <1>        ;sub   ah, ah
                             <1>        ;mov   [esi+IRQ.owner], ah ; 0 ; free !!!
                             <1>        ;and   byte [esi+IRQ.method], 80h
                             <1>        ;mov   [esi+IRQ.srb], ah ; 0
                             <1>        ;mov   [esi+IRQ.dev], ah ; 0
                             <1>        ;mov   dword [esi+IRQ.addr], 0
                             <1>        ;mov   dword [u.r0], 0
                             <1>
                             <1>        ;mov   byte [esi+IRQ.owner], 0
                             <1>
                             <1>        ; 22/04/2017
000108F1 29C0                <1>        sub    eax, eax
000108F3 8886[6E6B0100]      <1>        mov    [esi+IRQ.owner], al ; 0
                             <1>        ; 10/06/2017
000108F9 8686[806B0100]      <1>        xchg   al, [esi+IRQ.method]
000108FF 2480                <1>        and    al, 80h
00010901 745E                <1>        jz     short scbs_12
                             <1>        ; Audio device must be disabled -later- ! ([IRQ.medhod] = 80h)
                             <1>
                             <1> ;      cmp    byte [esi+IRQ.method], 80h ; device drv or kernel extension ?
                             <1> ;      jb     short scbs_12 ; bh = 0 reset to default IRQ handler
                             <1> ;
                             <1> ;      and    al, al
                             <1> ;      jz     short  scbs_5  ; the caller is 'syscalbac'
                             <1> ;      ; The caller is 'sysaudio' or ...
00010903 30C0                <1>        xor    al, al
                             <1> ;      mov    [esi+IRQ.method], al ; 0 ; reset kernel extension flag
                             <1> ;scbs_5:
                             <1> ;      sub    ah, ah
                             <1>        ;mov   [u.r0], eax ; 0
00010905 C3                  <1>        retn
                             <1>
                             <1> scbs_6:
                             <1>        ; 14/04/2017
00010906 20C0                <1>        and    al, al
00010908 7405                <1>        jz     short scbs_7  ; the caller is 'syscalbac'
                             <1>        ; AL = user number ([u.uno] or [audio.user] or ...)
                             <1>        ; The caller is 'sysaudio' or ...
                             <1>        ;
                             <1>        ; bh = method (0 = signal response byte, 1 = callback)
                             <1>        ;            (2 = auto increment of signal response byte)
                             <1>
0001090A 80CF80              <1>        or     bh, 80h            ; Kernel extension flag !
0001090D EB0A                <1>        jmp    short scbs_8
                             <1> scbs_7:
0001090F 8A86[806B0100]      <1>        mov    al, [esi+IRQ.method] ; >= 80h = kernel is using this IRQ
00010915 2480                <1>        and    al, 80h ; use only bit 7 (kernel function flag)
00010917 08C7                <1>        or     bh, al        ; method
                             <1>                             ; 0 = signal response byte, 1 = callback
                             <1>                             ; 2 = auto increment of s.r.b.
                             <1> scbs_8:
00010919 A0[B3030300]        <1>        mov    al, [u.uno] ; user (process) number (1 to 16)
0001091E 8886[6E6B0100]      <1>        mov    [esi+IRQ.owner], al ; lock the IRQ for user
00010924 88BE[806B0100]      <1>        mov    [esi+IRQ.method], bh
                             <1>
                             <1> ;      test   bh, 1
                             <1> ;      jnz    short scbs_9  ; Callback method, CX will not be used
                             <1> ;
                             <1> ;      test   bh, 2         ; use auto increment (counter) method
                             <1> ;      jz     short scbs_10 ; (count can be used for buffer switch)
                             <1> ;scbs_9:
                             <1> ;      xor    ecx, ecx ; 0
                             <1> scbs_10:
                             <1>        ;mov   [esi+IRQ.method], bh
0001092A 888E[896B0100]      <1>        mov    [esi+IRQ.srb], cl
00010930 C686[776B0100]00    <1>        mov    byte [esi+IRQ.dev], 0 ; device number is always 0
                             <1>                             ; for this system call
                             <1>        ;test  bh, 1
00010937 80E701              <1>        and    bh, 1 ; 17/04/2017
0001093A 7513                <1>        jnz    short scbs_11 ; callback method, use virtual address
                             <1>
0001093C 53                  <1>        push   ebx ; IRQ number (in BL)
0001093D 89D3                <1>        mov    ebx, edx
                             <1>        ; ebx = virtual address
                             <1>        ; [u.pgdir] = page directory's physical address
```

```
0001093F FE05[0E6B0100]       <1>       inc     byte [no_page_swap] ; 1
                              <1>                   ; Do not add this page to swap queue
                              <1>                   ; and remove it from swap queue if it is
                              <1>                   ; on the queue.
00010945 E84549FFFF           <1>       call   get_physical_addr
0001094A 5B                   <1>       pop    ebx
0001094B 728A                 <1>       jc     scbs_3 ; invalid address !
                              <1>       ; eax = physical address of the virtual address in user's space
0001094D 89C2                 <1>       mov    edx, eax
                              <1> scbs_11:
0001094F 66C1E602             <1>       shl    si, 2        ; byte (index) to dword (offset)
00010953 8996[926B0100]       <1>       mov    [esi+IRQ.addr], edx
                              <1>
00010959 FE05[D6030300]       <1>       inc    byte [u.irqc]; increase IRQ (in use) count
                              <1>
0001095F FEC7                 <1>       inc     bh ; 17/04/2017
                              <1>       ; bh > 0 -> set to requested IRQ handler (IRQ_u_list)
                              <1> scbs_12:
00010961 88D8                 <1>       mov    al, bl ; IRQ number
00010963 88FC                 <1>       mov    ah, bh ; 0 = reset, >0 = set
00010965 E8DAF3FFFF           <1>       call   set_hardware_int_vector
                              <1>
0001096A 31C0                 <1>       xor    eax, eax
                              <1>       ;mov   [u.r0], eax ; 0
                              <1>
0001096C C3                   <1>       retn   ; return with success (cf=0, eax=0)
                              <1>
                              <1>
                              <1> sysdma: ; DMA FUNCTIONS
                              <1>       ; 02/09/2017
                              <1>       ; 28/08/2017
                              <1>       ; 20/08/2017 - TRDOS 386 (TRDOS v2.0)
                              <1>       ;
                              <1>       ; Inputs:
                              <1>       ;     BH = 0 -> Allocate DMA buffer
                              <1>       ;        BL = 0 -> Use the system's default DMA
                              <1>       ;                 (SB16) Buffer
                              <1>       ;             Buffer Size (max.) = 65536 bytes
                              <1>       ;        BL > 0 -> Allocate (a new) DMA buffer
                              <1>       ;        ECX = DMA Buffer Size in bytes (<=128KB)
                              <1>       ;        EDX = Virtual Address of DMA buffer
                              <1>       ;
                              <1>       ;     BH = 1 -> Initialize (Start) DMA service
                              <1>       ;          BL, bit 0 to 3 = Channel Number (0 to 7)
                              <1>       ;          BL, bit 7 = Auto Initialized Mode
                              <1>       ;                  (If bit 7 is set)
                              <1>       ;             bit 6 = Record (read) mode (0= playback)
                              <1>       ;          ECX = byte count (0 = use dma buffer size)
                              <1>       ;          EDX = physical buffer address
                              <1>       ;               (0 = use dma buffer -start- address)
                              <1>       ;
                              <1>       ;     BH = 2 -> Get Current DMA Buffer Offset
                              <1>       ;          BL = DMA channel number
                              <1>       ;
                              <1>       ;     BH = 3 -> Get Current DMA count down value
                              <1>       ;          BL = DMA channel number (0 tO 7)
                              <1>       ;
                              <1>       ;     BH = 4 -> Get Current DMA channel (in progress)
                              <1>       ;
                              <1>       ;     BH = 5 -> Get System's Default DMA Buffer Address
                              <1>       ;
                              <1>       ;     BH = 6 -> Get Current DMA Buffer Address
                              <1>       ;
                              <1>       ;     BH = 7 -> Stop DMA service
                              <1>       ;
                              <1>       ;
                              <1>       ; Outputs:
                              <1>       ;
                              <1>       ;     For BH = 0 ; Allocate DMA buffer
                              <1>       ;         EAX = Physical address of DMA buffer
                              <1>       ;         ECX = Allocated buffer size in bytes
                              <1>       ;              - page count * 4096 -
                              <1>       ;              (may be bigger than requested)
                              <1>       ;         If BL input > 0,
                              <1>       ;            'sysalloc:' system call will be used with
                              <1>       ;            EBX (for 'sysalloc') = EDX (for 'sysdma')
                              <1>       ;            ECX is same, byte count (buffer size)
                              <1>       ;            EDX = 1024*1024*16 ; 16 MB upper limit
                              <1>       ;         If BL input = 0,
                              <1>       ;            Default DMA buffer (SB16 buffer) will be
                              <1>       ;            checked and if it is free, it's address
                              <1>       ;            will be returned in EAX and it's size
                              <1>       ;            will be returned in ECX (as 65536)
                              <1>       ;
                              <1>       ;         If CF = 1, error code is in EAX
                              <1>       ;            EAX = -1 ; DMA buffer allocation error!
                              <1>       ;            EAX = 11 ; 'Permission Denied' error !
                              <1>       ;
                              <1>       ;            Note: 'sysalloc' error return method
                              <1>       ;                  will be applied if BL input > 0 !
                              <1>       ;
                              <1>       ;      For BH = 1 ; Initialize (Start) DMA
                              <1>       ;          EAX = 0 (Successful)
                              <1>       ;          If CF = 1, error code is in EAX
                              <1>       ;
                              <1>       ;      For BH = 2 ; Get Current DMA Buffer Offset
                              <1>       ;          EAX = DMA Buffer Offset (in bytes)
                              <1>       ;          ;
                              <1>       ;           AX = DMA buffer offset
                              <1>       ;          EAX bits 16 to 23 = Page register value
                              <1>       ;
                              <1>       ;      For BH = 3 ; Get Current DMA count down value
                              <1>       ;          EAX = Count down value (remain bytes)
                              <1>       ;
                              <1>       ;      For BH = 4 ; Get Current DMA channel (in progress)
                              <1>       ;          EAX = DMA channel number (0 to 7)
                              <1>       ;            AH = 0 if the owner is the caller process
                              <1>       ;            AH > 0 if the dma channel is in use by
                              <1>       ;                   another user/process
                              <1>       ;          EAX = -1 (0FFFFFFFFh)
                              <1>       ;                if DMA service is not in use
                              <1>       ;                (stopped or not initialized/started)
                              <1>       ;
                              <1>       ;      For BH = 5 ; Get System's Default DMA Buff Addr
                              <1>       ;          EAX = Default DMA Buffer Address (Physical)
                              <1>       ;            = offset 'sb16_dma_buffer:'
                              <1>       ;          ECX = Buffer size
                              <1>       ;            = 65536
                              <1>       ;
                              <1>       ;      For BH = 6 ; Get Current DMA Buffer Address
                              <1>       ;          EAX = Current DMA buffer address (Physical)
                              <1>       ;          ECX = Current DMA buffer size (setting value)
                              <1>       ;          Note: These values are for current dma channel
                              <1>       ;               settings for the user/process
                              <1>       ;          ** For now (for current TRDOS 386 version)
                              <1>       ;            only one user/process can use only one
                              <1>       ;            dma channel & one dma buffer at same time
                              <1>       ;            (no multi tasking on DMA service) !!! **
                              <1>       ;          (Once, current DMA user must stop it's own DMA
                              <1>       ;           DMA service, than another user/program
                              <1>       ;           can use DMA service with same dma channel
                              <1>       ;           or with another DMA channel.)
                              <1>       ;
                              <1>       ;      For BH = 7 ; Stop DMA service (for current user
                              <1>       ;          and current DMA channel)
                              <1>       ;          EAX = 0 ; successful
                              <1>       ;          CF = 1 & EAX > 0 (= -1) -> Error
                              <1>
0001096D 80FF07               <1>       cmp    bh, 7
00010970 7612                 <1>       jna    short sysdma_0
                              <1>
                              <1> sysdma_err:
00010972 31C0                 <1>       xor    eax, eax
00010974 48                   <1>       dec    eax ; -1
                              <1> sysdma_perm_err:
00010975 A3[64030300]         <1>       mov    [u.r0], eax
0001097A A3[C8030300]         <1>       mov    [u.error], eax ; DMA service error !
0001097F E93ABDFFFF           <1>       jmp    error
                              <1>
                              <1> sysdma_0:
00010984 08FF                 <1>       or     bh, bh
00010986 0F85BA000000         <1>       jnz    sysdma_1
                              <1>
0001098C 20DB                 <1>       and    bl, bl
0001098E 7416                 <1>       jz     short sysdma_01
                              <1>
                              <1>       ; redirect system call to 'sysalloc'
00010990 89D3                 <1>       mov    ebx, edx ; virtual address of DMA buffer
                              <1>       ;ecx = Buffer size in bytes
                              <1>       ; DMA buffer address <= 16MB upper limit
00010992 BA00000001           <1>       mov    edx, 1024*1024*16 ; 16MB limit for DMA buff
                              <1>
00010997 C705[00700100]FFFF-  <1>       mov    dword [dma_addr], 0FFFFFFFFh ; -1
0001099F FFFF
                              <1>
000109A1 E9A8E5FFFF           <1>       jmp    sysalloc
                              <1>
                              <1> sysdma_01:
000109A6 B8[00000200]         <1>       mov    eax, sb16_dma_buffer
                              <1>
000109AB 803D[BD6B0100]01     <1>       cmp    byte [audio_device], 1
000109B2 722A                 <1>       jb     short sysdma_03
                              <1>
000109B4 3B05[DC6B0100]       <1>       cmp    eax, [audio_dma_buff]
000109BA 7507                 <1>       jne    short sysdma_02
                              <1>
                              <1> sysdma_0_err:
000109BC B80B000000           <1>       mov    eax, ERR_PERM_DENIED
000109C1 EBB2                 <1>       jmp    short sysdma_perm_err
                              <1>
                              <1> sysdma_02:
                              <1>       ; Only one user is permitted for audio/dma functions
                              <1>
000109C3 833D[DC6B0100]00     <1>       cmp    dword [audio_dma_buff], 0
000109CA 7612                 <1>       jna    short sysdma_03
                              <1>
000109CC 8A1D[E56B0100]       <1>       mov    bl, [audio_user]
000109D2 08DB                 <1>       or     bl, bl
000109D4 7408                 <1>       jz     short sysdma_03
                              <1>
000109D6 3A1D[B3030300]       <1>       cmp    bl, [u.uno]
000109DC 75DE                 <1>       jne    short sysdma_0_err
                              <1>
                              <1> sysdma_03:
000109DE 8A1D[FD6F0100]       <1>       mov    bl, [dma_user]
000109E4 20DB                 <1>       and    bl, bl
000109E6 750E                 <1>       jnz    short sysdma_04
                              <1>
000109E8 8A1D[B3030300]       <1>       mov    bl, [u.uno]
000109EE 881D[FD6F0100]       <1>       mov    [dma_user], bl
                              <1>
000109F4 EB15                 <1>       jmp    short sysdma_05
                              <1>
                              <1> sysdma_04:
000109F6 8B35[00700100]       <1>       mov    esi, [dma_addr]
000109FC 21F6                 <1>       and    esi, esi
```

```
000109FE 740B                <1>       jz     short sysdma_05
                             <1> 
00010A00 46                  <1>       inc    esi ; -1 -> 0
00010A01 7408                <1>       jz     short sysdma_05
                             <1> 
00010A03 3A1D[B3030300]      <1>       cmp    bl, [u.uno]
00010A09 75B1                <1>       jne    short sysdma_0_err
                             <1> 
                             <1> sysdma_05:
                             <1>       ; edx = virtual address (user's buffer address)
                             <1>       ;
00010A0B 81F900000100        <1>       cmp    ecx, 65536   ; byte count (buffer size)
00010A11 0F875BFFFFFF        <1>       ja     sysdma_err
                             <1>       ;
00010A17 81C1FF0F0000        <1>       add    ecx, PAGE_SIZE-1 ; 4095
00010A1D 6681E100F0          <1>       and    cx, ~PAGE_OFF ; not 4095
                             <1>       ;cmp   ecx, 65536
                             <1>       ;ja    sysdma_err ;
00010A22 51                  <1>       push   ecx    ; buffer size (allocated pages * 4096)
00010A23 50                  <1>       push   eax    ; offset sb16_dma_buffer
00010A24 89D3                <1>       mov    ebx, edx
00010A26 C1E90C              <1>       shr    ecx, 12 ; byte count to page count
                             <1>       ; eax = physical address of (audio) dma buffer
                             <1>       ; ebx = virtual address of (audio) dma buffer (user's pgdir)
                             <1>       ; ecx = page count (>0)
00010A29 E8734CFFFF          <1>       call   direct_memory_access
00010A2E 58                  <1>       pop    eax
00010A2F 59                  <1>       pop    ecx
00010A30 0F823CFFFFFF        <1>       jc     sysdma_err
                             <1> 
00010A36 A3[00700100]        <1>       mov    [dma_addr], eax
00010A3B 890D[04700100]      <1>       mov    [dma_size], ecx ; dma buffer size (in bytes)
                             <1> 
                             <1>       ;mov   [u.r0], eax ; DMA Buffer Address (Physical)
                             <1> 
                             <1>       ;mov   ebp, [u.usp]  ; ebp points to user's registers
                             <1>       ;mov   [ebp+24], ecx ; return to user with ecx value
                             <1> 
                             <1>       ;jmp   sysret
                             <1> 
                             <1>       ; 28/08/2017
00010A41 E9C4000000          <1>       jmp    sysdma_51
                             <1> 
                             <1> sysdma_1:
00010A46 80FF01              <1>       cmp    bh, 1
00010A49 0F87A6000000        <1>       ja     sysdma_5
                             <1> 
00010A4F F6C340              <1>       test   bl, 40h      ; record (read) mode -BL, bit 6-
00010A52 0F851AFFFFFF        <1>       jnz    sysdma_err ; not ready yet!
                             <1> 
00010A58 A1[00700100]        <1>       mov    eax, [dma_addr] ; physical address of dma buffer
00010A5D 21C0                <1>       and    eax, eax
00010A5F 0F840DFFFFFF        <1>       jz     sysdma_err
                             <1> 
00010A65 09D2                <1>       or     edx, edx
00010A67 7504                <1>       jnz    short sysdma_11
                             <1> 
00010A69 89C2                <1>       mov    edx, eax
00010A6B EB08                <1>       jmp    short sysdma_12
                             <1> sysdma_11:
00010A6D 39C2                <1>       cmp    edx, eax
00010A6F 0F82FDFEFFFF        <1>       jb     sysdma_err
                             <1> sysdma_12:
00010A75 21C9                <1>       and    ecx, ecx
00010A77 7508                <1>       jnz    short sysdma_13
                             <1> 
00010A79 8B0D[04700100]      <1>       mov    ecx, [dma_size]
00010A7F EB0C                <1>       jmp    short sysdma_14
                             <1> sysdma_13:
00010A81 3B0D[04700100]      <1>       cmp    ecx, [dma_size]
00010A87 0F87E5FEFFFF        <1>       ja     sysdma_err
                             <1> sysdma_14:
00010A8D 89C6                <1>       mov    esi, eax
00010A8F 0335[04700100]      <1>       add    esi, [dma_size]
                             <1> 
00010A95 89D0                <1>       mov    eax, edx
00010A97 01C8                <1>       add    eax, ecx
00010A99 0F82D3FEFFFF        <1>       jc     sysdma_err ; 02/09/2017
                             <1> 
00010A9F 39F0                <1>       cmp    eax, esi
00010AA1 0F87CBFEFFFF        <1>       ja     sysdma_err
                             <1> 
00010AA7 8B3D[DC6B0100]      <1>       mov    edi, [audio_dma_buff]
00010AAD 8B35[00700100]      <1>       mov    esi, [dma_addr]
                             <1> 
00010AB3 09FF                <1>       or     edi, edi
00010AB5 7424                <1>       jz     short sysdma_16
                             <1> 
00010AB7 803D[BD6B0100]01    <1>       cmp    byte [audio_device], 1
00010ABE 7208                <1>       jb     short sysdma_15
                             <1> 
                             <1>       ; Sound Blaster 16
00010AC0 39FE                <1>       cmp    esi, edi
00010AC2 0F84F4FEFFFF        <1>       je     sysdma_0_err ; permmission denied !
                             <1> 
                             <1> sysdma_15:
00010AC8 C605[FF6F0100]48    <1>       mov    byte [dma_mode], 48h ; single mode playback
                             <1> 
00010ACF F6C380              <1>       test   bl, 80h ; DMA mode - BL, bit 7, auto init -
00010AD2 7407                <1>       jz     short sysdma_16
                             <1>       ; Auto initialized playback (write) mode
00010AD4 8005[FF6F0100]10    <1>       add    byte [dma_mode], 10h ; = 58h
                             <1> sysdma_16:
```

```
00010ADB 80E307              <1>       and   bl, 07h
00010ADE 881D[FE6F0100]      <1>       mov   [dma_channel], bl
00010AE4 8915[08700100]      <1>       mov   [dma_start], edx
00010AEA 890D[0C700100]      <1>       mov   [dma_count], ecx
                             <1> 
                             <1>       ; 28/08/2017
                             <1>       ;call dma_init
                             <1>       ;jmp  sysret
00010AF0 E94B010000          <1>       jmp   dma_init
                             <1> 
                             <1> sysdma_5:
00010AF5 80FF05              <1>       cmp   bh, 5
00010AF8 7223                <1>       jb    short sysdma_3
00010AFA 0F87CE000000        <1>       ja    sysdma_6
                             <1> 
                             <1>       ; Get the system's default dma buffer addr and size
00010B00 B8[00000200]        <1>       mov   eax, sb16_dma_buffer
00010B05 B900000100          <1>       mov   ecx, 65536 ; Buffer size in bytes
                             <1> 
                             <1> sysdma_51:
                             <1>       ; 0 = there is not a dma buffer (in use or available)
00010B0A A3[64030300]        <1>       mov   [u.r0], eax
                             <1> 
00010B0F 8B2D[60030300]      <1>       mov   ebp, [u.usp]  ; ebp points to user's registers
00010B15 894D18              <1>       mov   [ebp+24], ecx ; return to user with ecx value
                             <1> 
00010B18 E9C1BBFFFF          <1>       jmp   sysret
                             <1> 
                             <1> sysdma_3:
00010B1D 80FF03              <1>       cmp   bh, 3
00010B20 7231                <1>       jb    short sysdma_2
00010B22 776B                <1>       ja    short sysdma_4
                             <1> 
                             <1>       ; Get current dma count down value (remain bytes)
                             <1>       ; 28/08/2017
00010B24 0FB635[FE6F0100]    <1>       movzx esi, byte [dma_channel]
00010B2B 0FB696[40160100]    <1>       movzx edx, byte [dma_flip+esi]
00010B32 EE                  <1>       out   dx, al              ; flip-flop clear
00010B33 8A96[20160100]      <1>       mov   dl, [dma_cnt+esi] ; dma count register addr
00010B39 EC                  <1>       in    al, dx
00010B3A 0FB6D8              <1>       movzx ebx, al
00010B3D EC                  <1>       in    al, dx
00010B3E 88C7                <1>       mov     bh, al
                             <1> 
00010B40 6683FE04            <1>       cmp   si, 4 ; channel number ?
00010B44 7202                <1>       jb    short sysdma_31 ; 8 bit dma channel
                             <1> 
00010B46 D1E3                <1>       shl   ebx, 1 ; word count to byte count
                             <1> 
                             <1> sysdma_31:
00010B48 891D[64030300]      <1>       mov   [u.r0], ebx
                             <1> 
00010B4E E98BBBFFFF          <1>       jmp   sysret
                             <1> 
                             <1> sysdma_2:
                             <1>       ; Get current dma buffer offset (& page)
                             <1>       ; 28/08/2017
00010B53 0FB635[FE6F0100]    <1>       movzx esi, byte [dma_channel]
00010B5A 0FB696[40160100]    <1>       movzx edx, byte [dma_flip+esi]
00010B61 EE                  <1>       out   dx, al              ; flip-flop clear
00010B62 8A96[18160100]      <1>       mov   dl, [dma_adr+esi]
00010B68 EC                  <1>       in    al, dx              ; get dma position
00010B69 0FB6D8              <1>       movzx ebx, al
00010B6C EC                  <1>       in    al, dx
00010B6D 88C7                <1>       mov   bh, al
                             <1> 
00010B6F 6683FE04            <1>       cmp   si, 4 ; channel number ?
00010B73 7202                <1>       jb    short sysdma_21 ; 8 bit dma channel
                             <1> 
00010B75 D1E3                <1>       shl   ebx, 1 ; word offset to byte offset
                             <1> 
                             <1> sysdma_21:
00010B77 891D[64030300]      <1>       mov   [u.r0], ebx
                             <1> 
00010B7D 8A96[28160100]      <1>       mov   dl, [dma_page+esi]
00010B83 EC                  <1>       in    al, dx              ; get dma page
                             <1> 
                             <1>       ;add  [u.ro+2], al
00010B84 0805[66030300]      <1>       or    [u.r0+2], al
                             <1> 
00010B8A E94FBBFFFF          <1>       jmp   sysret
                             <1> 
                             <1> sysdma_4:
                             <1>       ; Get current DMA channel number
                             <1>       ; 28/08/2017
00010B8F 8A25[FD6F0100]      <1>       mov   ah, [dma_user]
00010B95 20E4                <1>       and   ah, ah
00010B97 750F                <1>       jnz   short sysdma_42
                             <1> 
                             <1> sysdma_41:
                             <1>       ; Not a valid dma channel (in use)
00010B99 C705[64030300]FFFF- <1>       mov   dword [u.r0], -1 ; 0FFFFFFFFh
00010BA1 FFFF                <1>
00010BA3 E936BBFFFF          <1>       jmp   sysret
                             <1> 
                             <1> sysdma_42:
00010BA8 8B35[00700100]      <1>       mov   esi, [dma_addr]
00010BAE 21F6                <1>       and   esi, esi
00010BB0 74E7                <1>       jz    short sysdma_41
                             <1> 
00010BB2 46                  <1>       inc   esi ; -1 -> 0
00010BB3 74E4                <1>       jz    short sysdma_41
                             <1> 
```

```
00010BB5 A0[FE6F0100]        <1>       mov    al, [dma_channel]
                             <1> 
00010BBA 3A25[B3030300]      <1>       cmp    ah, [u.uno]
00010BC0 7502                <1>       jne    short sysdma_43
                             <1> 
00010BC2 30E4                <1>       xor    ah, ah ; DMA channel in use by current user
                             <1> 
                             <1> sysdma_43:
00010BC4 A3[64030300]        <1>       mov    [u.r0], eax ; AL = dma channel number
                             <1>                        ; AH > 0 if the the channel
                             <1>                        ; in use by another user/process
00010BC9 E910BBFFFF          <1>       jmp    sysret
                             <1> 
                             <1> sysdma_6:
00010BCE 80FF06              <1>       cmp    bh, 6
00010BD1 7710                <1>       ja     short sysdma_7
                             <1> 
                             <1>       ; 28/08/2017
                             <1>       ; Get current DMA buffer addr and size
00010BD3 A1[00700100]        <1>       mov    eax, [dma_addr] ; dma buffer address
00010BD8 8B0D[04700100]      <1>       mov    ecx, [dma_size] ; dma buffer size (in bytes)
                             <1> 
00010BDE E927FFFFFF          <1>       jmp    sysdma_51
                             <1> 
                             <1> sysdma_7:
                             <1>       ; DMA service STOP
00010BE3 A0[B3030300]        <1>       mov    al, [u.uno]
00010BE8 3A05[FD6F0100]      <1>       cmp    al, [dma_user]
00010BEE 751D                <1>       jne    short sysdma_72
                             <1> 
00010BF0 28C0                <1>       sub    al, al ; 0
                             <1> 
00010BF2 A2[FD6F0100]        <1>       mov    [dma_user], al ; clear user
                             <1> 
00010BF7 8605[FF6F0100]      <1>       xchg   al, [dma_mode]
00010BFD 20C0                <1>       and    al, al
                             <1>       ;jz    short sysdma_err
00010BFF 7527                <1>       jnz    short sysdma_73
                             <1> 
                             <1> sysdma_71:
00010C01 31C0                <1>       xor    eax, eax
00010C03 A3[64030300]        <1>       mov    [u.r0], eax; 0
00010C08 E9D1BAFFFF          <1>       jmp    sysret
                             <1> 
                             <1> sysdma_72:
                             <1>       ; 28/08/2017
00010C0D 803D[FD6F0100]00    <1>       cmp    byte [dma_user], 0
00010C14 76EB                <1>       jna    short sysdma_71 ; Nothing to do !
                             <1> 
00010C16 833D[00700100]00    <1>       cmp    dword [dma_addr], 0
00010C1D 0F8799FDFFFF        <1>       ja     sysdma_0_err
                             <1> 
00010C23 A2[FD6F0100]        <1>       mov    [dma_user], al ; reset to current user
                             <1> 
                             <1> sysdma_73:
                             <1>       ; 28/08/2017
00010C28 0FB635[FE6F0100]    <1>       movzx  esi, byte [dma_channel]
00010C2F 0FB696[30160100]    <1>       movzx  edx, byte [dma_mask+esi]
00010C36 A0[FE6F0100]        <1>       mov    al, [dma_channel]
00010C3B 0C04                <1>       or     al, 4
00010C3D EE                  <1>       out    dx, al
                             <1> 
00010C3E EBC1                <1>       jmp    short sysdma_71
                             <1> 
                             <1> dma_init:
                             <1>       ; 28/08/2017
                             <1>       ; 20/08/2017
                             <1>       ; DMA initialization
                             <1>       ; 14/08/2017
                             <1>       ; 03/08/2017, 06/08/2017, 08/08/2017
                             <1>       ; 02/07/2017, 13/07/2017, 16/07/2017, 30/07/2017
                             <1>       ; (Derived from 'DMA_INIT' procedure in SB16MOD.ASM)
                             <1>       ; Modified for TRDOS 386 DMA buffer allocation & initialization !
                             <1> 
00010C40 8B1D[08700100]      <1>       mov    ebx, [dma_start]
00010C46 8B0D[0C700100]      <1>       mov    ecx, [dma_count]
                             <1> 
00010C4C 0FB635[FE6F0100]    <1>       movzx  esi, byte [dma_channel]
                             <1> 
00010C53 6683FE04            <1>       cmp    si, 4
00010C57 7205                <1>       jb     short gdmi1
                             <1>       ; 08/08/2017
00010C59 66D1E9              <1>       shr    cx, 1 ; word count
00010C5C D1EB                <1>       shr    ebx, 1 ; convert byte offset to word offset
                             <1> gdmi1:
                             <1>       ;mov   [dma_poff], bx ; 08/08/2017
00010C5E 6649                <1>       dec    cx                 ; dma size = block size - 1
                             <1> 
00010C60 0FB696[30160100]    <1>       movzx  edx, byte [dma_mask+esi] ; 30/07/2017
00010C67 A0[FE6F0100]        <1>       mov    al, [dma_channel]
00010C6C 0C04                <1>       or     al, 4
00010C6E EE                  <1>       out    dx, al             ; dma channel mask
                             <1> 
00010C6F 30C0                <1>       xor    al, al ; 0 ; any value ! 08/08/2017
00010C71 8A96[40160100]      <1>       mov    dl, [dma_flip+esi]
00010C77 EE                  <1>       out    dx, al             ; flip-flop clear
                             <1> 
00010C78 8A96[38160100]      <1>       mov    dl, [dma_mod+esi]
00010C7E A0[FE6F0100]        <1>       mov    al, [dma_channel]  ; 13/07/2017
00010C83 2403                <1>       and    al, 3
                             <1>       ; 08/08/2017
00010C85 0A05[FF6F0100]      <1>       or     al, [dma_mode] ; 58h    ; dma mode for SB16
00010C8B EE                  <1>       out    dx, al
```

```
                              <1> 
00010C8C 8A96[18160100]       <1>       mov    dl, [dma_adr+esi]
00010C92 88D8                 <1>       mov    al, bl
00010C94 EE                   <1>       out    dx, al              ; offset low
                              <1> 
00010C95 88F8                 <1>       mov    al, bh
00010C97 EE                   <1>       out    dx, al              ; offset high
                              <1> 
00010C98 8A96[20160100]       <1>       mov    dl, [dma_cnt+esi]
00010C9E 88C8                 <1>       mov    al, cl
00010CA0 EE                   <1>       out    dx, al              ; size low
                              <1> 
00010CA1 88E8                 <1>       mov    al, ch
00010CA3 EE                   <1>       out    dx, al              ; size high
                              <1> 
00010CA4 8A96[28160100]       <1>       mov    dl, [dma_page+esi]
                              <1>       ; 14/08/2017
00010CAA 6683FE04             <1>       cmp    si, 4
00010CAE 7305                 <1>       jnb    short gdmi2
00010CB0 C1EB10               <1>       shr    ebx, 16
00010CB3 EB06                 <1>       jmp    short gdmi3
                              <1> gdmi2:
                              <1>       ; 09/08/2017
00010CB5 C1EB0F               <1>       shr    ebx, 15       ; complete 16 bit shift
00010CB8 80E3FE               <1>       and    bl, 0FEh ; clear bit 0 (not necessary)
                              <1> gdmi3:
00010CBB 88D8                 <1>       mov    al, bl
00010CBD EE                   <1>       out    dx, al              ; page
                              <1> 
00010CBE 8A96[30160100]       <1>       mov    dl, [dma_mask+esi]
00010CC4 A0[FE6F0100]         <1>       mov    al, [dma_channel]  ; 13/07/2017
00010CC9 2403                 <1>       and    al, 3
00010CCB EE                   <1>       out    dx, al              ; dma channel unmask
                              <1> 
                              <1>       ;retn
                              <1>       ; 28/08/2017
00010CCC E90DBAFFFF           <1>       jmp    sysret
                              <1> 
                              <1> otty:
                              <1> sret:
                              <1> ocvt:
                              <1> ctty:
                              <1> cret:
                              <1> ccvt:
                              <1> rtty:
                              <1> wtty:
                              <1> rmem:
                              <1> wmem:
                              <1> rfd:
                              <1> rhd:
                              <1> wfd:
                              <1> whd:
                              <1> rlpt:
                              <1> wlpt:
                              <1> rcvt:
                              <1> xmtt:
00010CD1 C3                   <1>       retn
                                  %include 'trdosk9.s' ; 04/01/2016
                              <1> ; ****************************************************************************
                              <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - INITIALIZED DATA : trdosk9.s
                              <1> ; ----------------------------------------------------------------------------
                              <1> ; Last Update: 31/12/2017
                              <1> ; ----------------------------------------------------------------------------
                              <1> ; Beginning: 04/01/2016
                              <1> ; ----------------------------------------------------------------------------
                              <1> ; ----------------------------------------------------------------------------
                              <1> ; Assembler: NASM version 2.11 (trdos386.s)
                              <1> ; ----------------------------------------------------------------------------
                              <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
                              <1> ; TRDOS2.ASM (09/11/2011)
                              <1> ; ****************************************************************************
                              <1> ; DRV_INIT.ASM [26/09/2009] Last Update: 07/08/2011
                              <1> ; MAINPROG.ASM [17/01/2004] Last Update: 09/11/2011
                              <1> ; CMD_INTR.ASM [29/01/2005] Last Update: 09/11/2011
                              <1> ; FILE.ASM [29/10/2009] Last Update: 09/10/2011
                              <1> 
                              <1> ; 12/02/2016
                              <1> Last_DOS_DiskNo:
00010CD2 01                   <1>             db 1 ; A: = 0 & B: = 1
                              <1> 
                              <1> Restore_CDIR:
00010CD3 FF                   <1>             db 0FFh ; Initial value -> any number except 0
                              <1> 
                              <1> msg_CRLF_temp:
00010CD4 070D0A00             <1>             db 07h, 0Dh, 0Ah, 0
                              <1> 
                              <1> Magic_Bytes:
00010CD8 04                   <1>             db 4
00010CD9 01                   <1>             db 1
                              <1> mainprog_Version:
00010CDA 07                   <1>             db 7
00010CDB 5B5452444F535D204D-  <1>             db "[TRDOS] Main Program v2.0.311217"
00010CE4 61696E2050726F6772-  <1>
00010CED 616D2076322E302E33-  <1>
00010CF6 3131323137           <1>
00010CFB 0D0A                 <1>             db 0Dh, 0Ah
00010CFD 286329204572646F67-  <1>             db "(c) Erdogan Tan 2005-2017"
00010D06 616E2054616E203230-  <1>
00010D0F 30352D32303137       <1>
00010D16 0D0A00               <1>             db 0Dh, 0Ah, 0
                              <1> 
                              <1> MainProgCfgFile: ; 14/04/2016
00010D19 4D41494E50524F472E-  <1>             db "MAINPROG.CFG", 0
```

```
00010D22 43464700             <1>
                              <1>
                              <1> TRDOSPromptLabel:
00010D26 5452444F53           <1>             db "TRDOS"
00010D2B 00                   <1>             db 0
00010D2C 00<rept>             <1>                 times 5 db 0
00010D31 00                   <1>             db 0
                              <1>
                              <1> ; INTERNAL COMMANDS
                              <1> Command_List:
00010D32 44495200             <1> Cmd_Dir:    db "DIR", 0
00010D36 434400               <1> Cmd_Cd:           db "CD", 0
00010D39 433A00               <1> Cmd_Drive:  db "C:", 0
00010D3C 56455200             <1> Cmd_Ver:    db "VER", 0
00010D40 4558495400           <1> Cmd_Exit:   db "EXIT", 0
00010D45 50524F4D505400       <1> Cmd_Prompt: db "PROMPT", 0
00010D4C 564F4C554D4500       <1> Cmd_Volume: db "VOLUME", 0
00010D53 4C4F4E474E414D4500   <1> Cmd_LongName:     db "LONGNAME", 0
00010D5C 4441544500           <1> Cmd_Date:   db "DATE", 0
00010D61 54494D4500           <1> Cmd_Time:   db "TIME", 0
00010D66 52554E00             <1> Cmd_Run:    db "RUN", 0
00010D6A 53455400             <1> Cmd_Set:    db "SET", 0
00010D6E 434C5300             <1> Cmd_Cls:    db "CLS", 0
00010D72 53484F5700           <1> Cmd_Show:   db "SHOW", 0
00010D77 44454C00             <1> Cmd_Del:    db "DEL", 0
00010D7B 41545452494200       <1> Cmd_Attrib: db "ATTRIB", 0
00010D82 52454E414D4500       <1> Cmd_Rename: db "RENAME", 0
00010D89 524D44495200         <1> Cmd_Rmdir:  db "RMDIR", 0
00010D8F 4D4B44495200         <1> Cmd_Mkdir:  db "MKDIR", 0
00010D95 434F505900           <1> Cmd_Copy:   db "COPY", 0
00010D9A 4D4F564500           <1> Cmd_Move:   db "MOVE", 0
00010D9F 5041544800           <1> Cmd_Path:   db "PATH", 0
00010DA4 4D454D00             <1> Cmd_Mem:    db "MEM", 0
00010DA8 00                   <1>             db 0
00010DA9 46494E4400           <1> Cmd_Find:   db "FIND", 0
00010DAE 4543484F00           <1> Cmd_Echo:   db "ECHO", 0
00010DB3 2A00                 <1> Cmd_Remark: db "*", 0
00010DB5 3F00                 <1> Cmd_Help:   db "?", 0
00010DB7 44455649434500       <1> Cmd_Device: db "DEVICE", 0
00010DBE 4445564C49535400     <1> Cmd_DevList: db "DEVLIST", 0
00010DC6 434844495200         <1> Cmd_Chdir:  db "CHDIR", 0
00010DCC 4245455000           <1> Cmd_Beep:   db "BEEP", 0
                              <1>
00010DD1 00                   <1>             db 0
                              <1>
                              <1> ; 15/02/2016 (FILE.ASM, 09/10/2011)
                              <1> invalid_fname_chars:
00010DD2 222728292A2B2C2F     <1>             db 22h, 27h, 28h, 29h, 2Ah, 2Bh, 2Ch, 2Fh
00010DDA 3A3B3C3D3E3F40       <1>             db 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh, 40h
00010DE1 5B5C5D5E60           <1>             db 5Bh, 5Ch, 5Dh, 5Eh, 60h
                              <1> sizeInvFnChars  equ ($ - invalid_fname_chars)
                              <1> ;
                              <1>
                              <1> Msg_Enter_Date:
00010DE6 456E746572206E6577-  <1>                 db 'Enter new date (dd-mm-yy): '
00010DEF 206461746520286464-  <1>
00010DF8 2D6D6D2D7979293A20   <1>
00010E01 00                   <1>                 db 0
                              <1> Msg_Show_Date:
00010E02 43757272656E742064-  <1>                 db   'Current date is '
00010E0B 61746520697320       <1>
00010E12 30                   <1> Day:            db   '0'
00010E13 30                   <1>             db   '0'
00010E14 2F                   <1>                 db   '/'
00010E15 30                   <1> Month:          db   '0'
00010E16 30                   <1>             db   '0'
00010E17 2F                   <1>                 db   '/'
00010E18 30                   <1> Century:        db   '0'
00010E19 30                   <1>                 db   '0'
00010E1A 30                   <1> Year:           db   '0'
00010E1B 30                   <1>             db   '0'
00010E1C 0D0A00               <1>                 db   0Dh, 0Ah, 0
                              <1>
                              <1> Msg_Enter_Time:
00010E1F 456E746572206E6577-  <1>             db 'Enter new time: '
00010E28 2074696D653A20       <1>
00010E2F 00                   <1>             db 0
                              <1> Msg_Show_Time:
00010E30 43757272656E742074-  <1>             db   'Current time is '
00010E39 696D6520697320       <1>
00010E40 30                   <1> Hour:           db   '0'
00010E41 30                   <1>             db   '0'
00010E42 3A                   <1>             db   ':'
00010E43 30                   <1> Minute:         db   '0'
00010E44 30                   <1>             db   '0'
00010E45 3A                   <1>             db   ':'
00010E46 30                   <1> Second:         db   '0'
00010E47 30                   <1>             db   '0'
00010E48 0D0A00               <1>             db   0Dh, 0Ah, 0
                              <1>
                              <1> ;VolSize_Unit1:   dd 0
                              <1> ;VolSize_Unit2:   dd 0
                              <1>
                              <1> VolSize_KiloBytes:
00010E4B 206B696C6F62797465-  <1>             db " kilobytes", 0Dh, 0Ah, 0
00010E54 730D0A00             <1>
                              <1> VolSize_Bytes:
00010E58 2062797465730D0A00   <1>             db " bytes", 0Dh, 0Ah, 0
                              <1> Volume_in_drive:
00010E61 0D0A                 <1>             db 0Dh, 0Ah
                              <1> Vol_FS_Name:
00010E63 54522046533120       <1>             db "TR FS1 "
00010E6A 566F6C756D6520696E-  <1>             db "Volume in drive "
```

```
00010E73 20647269766520      <1>
00010E7A 30                  <1> Vol_Drv_Name:   db 30h
00010E7B 3A                  <1>             db ":"
00010E7C 20697320            <1>             db " is "
00010E80 0D0A00              <1>             db 0Dh, 0Ah, 0
                             <1> Dir_Drive_Str:
00010E83 54522D444F53204472- <1>                 db "TR-DOS Drive "
00010E8C 69766520            <1>
                             <1> Dir_Drive_Name:
00010E90 303A                <1>                 db "0:"
00010E92 0D0A                <1>                 db  0Dh, 0Ah
                             <1> Vol_Str_Header:
00010E94 566F6C756D65204E61- <1>                 db "Volume Name: "
00010E9D 6D653A20            <1>
                             <1> Vol_Name:
00010EA1 00<rept>            <1>             times 64 db 0
00010EE1 00                  <1>             db 0
                             <1> Vol_Serial_Header:
00010EE2 0D0A                <1>             db 0Dh, 0Ah
00010EE4 566F6C756D65205365- <1>             db "Volume Serial No: "
00010EED 7269616C204E6F3A20  <1>
                             <1> Vol_Serial1:
00010EF6 30303030            <1>             db "0000"
00010EFA 2D                  <1>             db "-"
                             <1> Vol_Serial2:
00010EFB 30303030            <1>             db "0000"
00010EFF 0D0A00              <1>             db 0Dh, 0Ah, 0
                             <1>
                             <1> ;Vol_Tot_Sec_Str_Start:
                             <1> ;           dd 0
                             <1> Vol_Total_Sector_Header:
00010F02 0D0A                <1>             db 0Dh, 0Ah
00010F04 566F6C756D65205369- <1>             db "Volume Size : ", 0
00010F0D 7A65203A2000        <1>
                             <1> ;Vol_Tot_Sec_Str:
                             <1> ;           db "0000000000"
                             <1> ;Vol_Tot_Sec_Str_End:
                             <1> ;           db 0
                             <1> ;Vol_Free_Sectors_Str_Start:
                             <1> ;           dd 0
                             <1> Vol_Free_Sectors_Header:
00010F13 467265652053706163- <1>             db "Free Space  : ", 0
00010F1C 6520203A2000        <1>
                             <1> ;Vol_Free_Sectors_Str:
                             <1> ;           db "0000000000"
                             <1> ;Vol_Free_Sectors_Str_End:
                             <1> ;           db 0
                             <1>
                             <1> Dir_Str_Header:
00010F22 4469726563746F7279- <1>                 db "Directory: "
00010F2B 3A20                <1>
00010F2D 2F                  <1> Dir_Str_Root:   db "/"
00010F2E 00<rept>            <1> Dir_Str:        times 64 db 0
00010F6E 00000000            <1>                 dd 0
00010F72 00                  <1>                 db 0
                             <1>
                             <1> Msg_Bad_Command:
00010F73 42616420636F6D6D61- <1>                 db "Bad command or file name!"
00010F7C 6E64206F722066696C- <1>
00010F85 65206E616D6521      <1>
00010F8C 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1>
                             <1> msgl_drv_not_ready:
00010F8F 070D0A              <1>             db 07h, 0Dh, 0Ah
                             <1>
                             <1> ; CMD_INTR.ASM - 09/11/2011 - Messages
                             <1>
                             <1> Msg_Not_Ready_Read_Err:
00010F92 4472697665206E6F74- <1>                 db "Drive not ready or read error!"
00010F9B 207265616479206F72- <1>
00010FA4 207265616420657272- <1>
00010FAD 6F7221              <1>
00010FB0 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1>
                             <1> Msg_Not_Ready_Write_Err:
00010FB3 4472697665206E6F74- <1>                 db "Drive not ready or write error!"
00010FBC 207265616479206F72- <1>
00010FC5 207772697465206572- <1>
00010FCE 726F7221            <1>
00010FD2 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1>
                             <1> Msg_Dir_Not_Found:
00010FD5 4469726563746F7279- <1>                 db "Directory not found!"
00010FDE 206E6F7420666F756E- <1>
00010FE7 6421                <1>
00010FE9 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1>
                             <1> Msg_File_Not_Found:
00010FEC 46696C65206E6F7420- <1>                 db "File not found!"
00010FF5 666F756E6421        <1>
00010FFB 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1>
                             <1> Msg_File_Directory_Not_Found:
00010FFE 46696C65206F722064- <1>                 db "File or directory not found!"
00011007 69726563746F727920- <1>
00011010 6E6F7420666F756E64- <1>
00011019 21                  <1>
0001101A 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1>
                             <1> Msg_LongName_Not_Found:
0001101D 4C6F6E67206E616D65- <1>                 db "Long name not found!"
00011026 206E6F7420666F756E- <1>
0001102F 6421                <1>
```

```
00011031 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1> 
                             <1> beep_Insufficient_Memory: ; 20/02/2017
00011034 0D0A                <1>             db 0Dh, 0Ah
00011036 07                  <1>             db 07h
                             <1> Msg_Insufficient_Memory:
00011037 496E73756666696369- <1>                 db "Insufficient memory!"
00011040 656E74206D656D6F72- <1>
00011049 7921                <1>
0001104B 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1> 
                             <1> Msg_Error_Code:
0001104E 436F6D6D616E642066- <1>                 db 'Command failed! Error code : '
00011057 61696C656421204572- <1>
00011060 726F7220636F646520- <1>
00011069 3A20                <1>
0001106B 303068              <1> error_code_hex: db '00h'
0001106E 0A0A00              <1>                 db 0Ah, 0Ah, 0
                             <1> 
00011071 90                  <1> align 2
                             <1> 
                             <1> ; 10/02/2016
                             <1> ; DIR.ASM - 09/10/2011
                             <1> 
00011072 3C4449523E20202020- <1> Type_Dir:       db '<DIR>     ' ; 10 bytes
0001107B 20                  <1>
                             <1> 
                             <1> File_Name:
0001107C 20<rept>            <1>                 times 12 db 20h
00011088 20                  <1>             db 20h
                             <1> Dir_Or_FileSize:
00011089 20<rept>            <1>                 times 10 db 20h
00011093 20                  <1>             db 20h
                             <1> File_Attribute:
00011094 20202020            <1>             dd 20202020h
00011098 20                  <1>             db 20h
                             <1> File_Day:
00011099 3030                <1>                 db '0','0'
0001109B 2F                  <1>             db '/'
                             <1> File_Month:
0001109C 3030                <1>                 db '0','0'
0001109E 2F                  <1>             db '/'
                             <1> File_Year:
0001109F 30303030            <1>                 db '0','0','0','0'
000110A3 20                  <1>             db 20h
                             <1> File_Hour:
000110A4 3030                <1>                 db '0','0'
000110A6 3A                  <1>             db ':'
                             <1> File_Minute:
000110A7 3030                <1>                 db '0','0'
000110A9 00                  <1>             db 0
                             <1> 
                             <1> Decimal_File_Count_Header:
000110AA 0D0A                <1>             db 0Dh, 0Ah
                             <1> Decimal_File_Count:
000110AC 00<rept>            <1>             times 6 db 0
                             <1> 
000110B2 2066696C6528732920- <1> str_files:  db " file(s) & "
000110BB 2620                <1>
                             <1> Decimal_Dir_Count:
000110BD 00<rept>            <1>             times 6 db 0
                             <1> str_dirs:
000110C3 206469726563746F72- <1>             db " directory(s) "
000110CC 7928732920          <1>
000110D1 0D0A00              <1>             db 0Dh, 0Ah, 0
                             <1> 
000110D4 206279746528732920- <1> str_bytes:  db " byte(s) in file(s)"
000110DD 696E2066696C652873- <1>
000110E6 29                  <1>
000110E7 0D0A00              <1>             db 0Dh, 0Ah, 0
                             <1> 
                             <1> ; CMD_INTR.ASM - 09/11/2011
                             <1> ; 07/10/2010
                             <1> Msg_invalid_name_chars:
000110EA 496E76616C69642066- <1>                 db "Invalid file or directory name characters!"
000110F3 696C65206F72206469- <1>
000110FC 726563746F7279206E- <1>
00011105 616D65206368617261- <1>
0001110E 637465727321        <1>
00011114 0D0A00              <1>             db 0Dh, 0Ah, 0
                             <1> ; 21/02/2016
00011117 46696C65206F722064- <1> Msg_Name_Exists: db "File or directory name exists!"
00011120 69726563746F727920- <1>
00011129 6E616D652065786973- <1>
00011132 747321              <1>
00011135 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1> Msg_DoYouWantMkdir:
00011138 446F20796F75207761- <1>                 db "Do you want to make directory ", 0
00011141 6E7420746F206D616B- <1>
0001114A 65206469726563746F- <1>
00011153 72792000            <1>
00011157 2028592F4E29203F20- <1> Msg_YesNo:      db " (Y/N) ? ", 0
00011160 00                  <1>
00011161 000D0A00            <1> Y_N_nextline:      db 0, 0Dh, 0Ah, 0
00011165 4F4B2E0D0A00        <1> Msg_OK:            db "OK.", 0Dh, 0Ah, 0
                             <1> 
                             <1> ; 27/02/2016
                             <1> Msg_DoYouWantRmDir:
0001116B 446F20796F75207761- <1>                 db "Do you want to delete directory ", 0
00011174 6E7420746F2064656C- <1>
0001117D 657465206469726563- <1>
00011186 746F72792000        <1>
                             <1> Msg_Dir_Not_Empty:
```

```
0001118C 4469726563746F7279- <1>                 db "Directory not empty!"
00011195 206E6F7420656D7074- <1>
0001119E 7921                <1>
000111A0 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1>
                             <1> Msg_DoYouWantDelete:
000111A3 446F20796F75207761- <1>                 db "Do you want to delete file ",0
000111AC 6E7420746F2064656C- <1>
000111B5 6574652066696C6520- <1>
000111BE 00                  <1>
                             <1>
000111BF 44656C657465642E2E- <1> Msg_Deleted:    db "Deleted...", 0Dh, 0Ah, 0
000111C8 2E0D0A00            <1>
                             <1>
                             <1> Msg_Permission_Denied:
000111CC 07                  <1>                 db 7
000111CD 5065726D697373696F- <1>                 db "Permission denied!", 0Dh, 0Ah, 0
000111D6 6E2064656E69656421- <1>
000111DF 0D0A00              <1>
                             <1>
                             <1> ; 04/03/2016
000111E2 4E657720            <1> Msg_New:        db "New "
000111E6 00                  <1>                 db 0
                             <1> Str_Attributes:
000111E7 417474726962757465- <1>                 db "Attributes : "
000111F0 73203A20            <1>
000111F4 4E4F524D414C        <1> Attr_Chars:     db "NORMAL"
000111FA 00                  <1>                 db 0
                             <1>
                             <1> ; 06/03/2016
                             <1> ; CMD_INTR.ASM - 16/11/2010
                             <1> Msg_DoYouWantRename:
000111FB 446F20796F75207761- <1>                 db "Do you want to rename ", 0
00011204 6E7420746F2072656E- <1>
0001120D 616D652000          <1>
00011212 66696C652000        <1> Rename_File:    db "file ", 0
00011218 6469726563746F7279- <1> Rename_Directory: db "directory ", 0
00011221 2000                <1>
00011223 00<rept>            <1> Rename_OldName: times 13 db 0
00011230 20617320            <1> Msg_File_rename_as: db " as "
00011234 00<rept>            <1> Rename_NewName: times 13 db 0
                             <1>
                             <1> ; 08/03/2016
                             <1> ; CMD_INTR.ASM - 01/08/2010 - 23/04/2011
                             <1> msg_not_same_drv:
00011241 4E6F742073616D6520- <1>                 db "Not same drive!"
0001124A 647269766521        <1>
00011250 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1>
                             <1> Msg_DoYouWantMoveFile:
00011253 446F20796F75207761- <1>                 db "Do you want to move file", 0
0001125C 6E7420746F206D6F76- <1>
00011265 652066696C6500      <1>
                             <1>
                             <1> msg_insufficient_disk_space:
0001126C 496E73756666696369- <1>                 db "Insufficient disk space!"
00011275 656E74206469736B20- <1>
0001127E 737061636521        <1>
00011284 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1>
                             <1> ; 01/08/2010
                             <1> msg_source_file:
00011287 0D0A536F7572636520- <1>             db 0Dh, 0Ah, "Source file name      :   "
00011290 66696C65206E616D65- <1>
00011299 2020202020203A2020- <1>
000112A2 20                  <1>
                             <1> msg_source_file_drv:
000112A3 203A00              <1>             db " :", 0
                             <1> msg_destination_file:
000112A6 0D0A44657374696E61- <1>             db 0Dh, 0Ah, "Destination file name :   "
000112AF 74696F6E2066696C65- <1>
000112B8 206E616D65203A2020- <1>
000112C1 20                  <1>
                             <1> msg_destination_file_drv:
000112C2 203A00              <1>             db " :", 0
                             <1> msg_copy_nextline:
000112C5 0D0A00              <1>             db 0Dh, 0Ah, 0
                             <1>
                             <1> ; 15/03/2016
                             <1> ; CMD_INTR.ASM
                             <1>
                             <1> Msg_DoYouWantOverWriteFile:
000112C8 446F20796F75207761- <1>                 db "Do you want to overwrite file ",0
000112D1 6E7420746F206F7665- <1>
000112DA 727772697465206669- <1>
000112E3 6C652000            <1>
                             <1>
                             <1> Msg_DoYouWantCopyFile:
000112E7 446F20796F75207761- <1>                 db "Do you want to copy file",0
000112F0 6E7420746F20636F70- <1>
000112F9 792066696C6500      <1>
                             <1>
                             <1> Msg_read_file_error_before_EOF:
00011300 46696C652072656164- <1>             db "File reading error! (before EOF)"
00011309 696E67206572726F72- <1>
00011312 2120286265666F7265- <1>
0001131B 20454F4629          <1>
00011320 0A0A00              <1>             db 0Ah, 0Ah, 0
                             <1>
                             <1> ; 18/03/2016
                             <1> ; TRDOS 386 (v2.0) mainprog copy procedure
                             <1> msg_reading:
00011323 52656164696E672E2E- <1>             db "Reading... ", 0
```

```
0001132C 2E2000              <1>
                             <1> msg_writing:
0001132F 57726974696E672E2E- <1>              db "Writing... ", 0
00011338 2E2000              <1>
                             <1> percentagestr:
0001133B 2020202500          <1>              db "   %", 0  ; "  0%" .. "100%"
                             <1> ; 11/04/2016
                             <1> Msg_No_Set_Space:
00011340 496E73756666696369- <1>                 db "Insufficient environment space!"
00011349 656E7420656E766972- <1>
00011352 6F6E6D656E74207370- <1>
0001135B 61636521            <1>
0001135F 0D0A00              <1>                 db 0Dh, 0Ah, 0
                             <1> ; 18/04/2016
                             <1> isc_msg:
00011362 0D0A                <1>              db 0Dh, 0Ah
00011364 494E56414C49442053- <1>              db "INVALID SYSTEM CALL", 0
0001136D 595354454D2043414C- <1>
00011376 4C00                <1>
                             <1> usi_msg:
00011378 0D0A                <1>              db 0Dh, 0Ah
0001137A 554E444546494E4544- <1>              db "UNDEFINED SOFTWARE INTERRUPT", 0
00011383 20534F465457415245- <1>
0001138C 20494E544552525550- <1>
00011395 5400                <1>
                             <1> ifc_msg:
00011397 0D0A                <1>              db 0Dh, 0Ah
00011399 494E56414C49442046- <1>              db "INVALID FUNCTION CALL"
000113A2 554E4354494F4E2043- <1>
000113AB 414C4C              <1>
                             <1> inv_msg_for_trdos_v2:
000113AE 20                  <1>              db 20h
000113AF 666F72205452444F53- <1>              db "for TRDOS v2!"
000113B8 20763221            <1>
000113BC 07                  <1>              db 07h
000113BD 0D0A                <1>              db 0Dh, 0Ah
000113BF 0D0A                <1>              db 0Dh, 0Ah
000113C1 494E5420            <1>              db "INT "
000113C5 303068              <1> int_num_str: db "00h"
000113C8 0D0A                <1>              db 0Dh, 0Ah
000113CA 454158203A20        <1>              db "EAX : "
000113D0 303030303030303068- <1> eax_str:     db "00000000h", 0Dh, 0Ah
000113D9 0D0A                <1>
000113DB 454950203A20        <1>              db "EIP : "
000113E1 303030303030303068- <1> eip_str:     db "00000000h", 0Dh, 0Ah, 0
000113EA 0D0A00              <1>
                             <1>
                             <1> ; 07/10/2016
                             <1> ; Device names & parameters (for kernel devices)
                             <1>
000113ED 90                  <1> align 2
                             <1> KDEV_NAME:
000113EE 5454590000000000    <1>              db 'TTY',0,0,0,0,0 ; 1
000113F6 4D454D0000000000    <1>              db 'MEM',0,0,0,0,0 ; 2
000113FE 4644300000000000    <1>              db 'FD0',0,0,0,0,0 ; 3
00011406 4644310000000000    <1>              db 'FD1',0,0,0,0,0 ; 4
0001140E 4844300000000000    <1>              db 'HD0',0,0,0,0,0 ; 5
00011416 4844310000000000    <1>              db 'HD1',0,0,0,0,0 ; 6
0001141E 4844320000000000    <1>              db 'HD2',0,0,0,0,0 ; 7
00011426 4844330000000000    <1>              db 'HD3',0,0,0,0,0 ; 8
0001142E 4C50540000000000    <1>              db 'LPT',0,0,0,0,0 ; 9
00011436 5454593000000000    <1>              db 'TTY0',0,0,0,0 ; 10
0001143E 5454593100000000    <1>              db 'TTY1',0,0,0,0 ; 11
00011446 5454593200000000    <1>              db 'TTY2',0,0,0,0 ; 12
0001144E 5454593300000000    <1>              db 'TTY3',0,0,0,0 ; 13
00011456 5454593400000000    <1>              db 'TTY4',0,0,0,0 ; 14
0001145E 5454593500000000    <1>              db 'TTY5',0,0,0,0 ; 15
00011466 5454593600000000    <1>              db 'TTY6',0,0,0,0 ; 16
0001146E 5454593700000000    <1>              db 'TTY7',0,0,0,0 ; 17
00011476 5454593800000000    <1>              db 'TTY8',0,0,0,0 ; 18
0001147E 5454593900000000    <1>              db 'TTY9',0,0,0,0 ; 19
00011486 434F4D3100000000    <1>              db 'COM1',0,0,0,0 ; 18
0001148E 434F4D3200000000    <1>              db 'COM2',0,0,0,0 ; 19
                             <1>              ;db 'CONSOLE',0       ; 1
                             <1>              ;db 'PRINTER',0   ; 9
                             <1>              ;db 'CDROM'    ; 20
                             <1>              ;db 'CDROM0'   ; 20
                             <1>              ;db 'CDROM1'   ; 21
                             <1>              ;db 'DVD'      ; 22
                             <1>              ;db 'DVD0'     ; 22
                             <1>              ;db 'DVD1'     ; 23
                             <1>              ;db 'USB'      ; 24
                             <1>              ;db 'USB0'     ; 24
                             <1>              ;db 'USB1'     ; 25
                             <1>              ;db 'USB2'     ; 26
                             <1>              ;db 'USB3'        ; 27
                             <1>              ;db 'KEYBOARD'       ; 1
                             <1>              ;db 'MOUSE'    ; 28
                             <1>              ;db 'SOUND'    ; 29
                             <1>              ;db 'VGA',0,0,0,0 ; 30
                             <1>              ;db 'CGA',0,0,0,0 ; 31
                             <1>              ;db 'AUDIO',0,0,0 ; 29
                             <1>              ;db 'VIDEO',0,0,0 ; 32
                             <1>              ;db 'MUSIC',0,0,0 ; 33
                             <1>              ;db 'ETHERNET'       ; 34
                             <1>              ;db 'SD0',0,0,0,0,0 ; 35
                             <1>              ;db 'SD1',0,0,0,0,0 ; 36
                             <1>              ;db 'SD2',0,0,0,0,0 ; 37
                             <1>              ;db 'SD3',0,0,0,0,0 ; 38
                             <1>              ;db 'SATA0'    ; 35
                             <1>              ;db 'SATA1'    ; 36
                             <1>              ;db 'SATA2'        ; 37
                             <1>              ;db 'SATA3'        ; 38
                             <1>             ;db 'PATA0',0,0,0  ; 5
                             <1>             ;db 'PATA1',0,0,0  ; 6
                             <1>             ;db 'PATA2',0,0,0  ; 7
                             <1>             ;db 'PATA3',0,0,0  ; 8
                             <1>             ;db 'WIRELESS'        ; 39
                             <1>             ;db 'HDMI',0,0,0,0 ; 40
00011496 4E554C4C00000000    <1>             db 'NULL',0,0,0,0 ; 0
                             <1>
                             <1> NumOfKernelDevNames equ ($-KDEV_NAME) / 8 ; 20 (07/10/2016)
                             <1>
                             <1> KDEV_NUMBER:
0001149E 010203040506070809  <1>             db 1,2,3,4,5,6,7,8,9
000114A7 0A0B0C0D0E0F101112- <1>             db 10,11,12,13,14,15,16,17,18,19
000114B0 13                  <1>
000114B1 121300              <1>             db 18,19,0
                             <1>
                             <1> NumOfKernelDevices equ $ - KDEV_NUMBER
                             <1>
                             <1> KDEV_OADDR:
000114B4 [D10C0100]          <1>             dd otty ;tty  ; 1
000114B8 [D10C0100]          <1>             dd sret ;mem  ; 2
000114BC [D10C0100]          <1>             dd sret ;fd0  ; 3
000114C0 [D10C0100]          <1>             dd sret ;fd1  ; 4
000114C4 [D10C0100]          <1>             dd sret ;hd0  ; 5
000114C8 [D10C0100]          <1>             dd sret ;hd1  ; 6
000114CC [D10C0100]          <1>             dd sret ;hd2  ; 7
000114D0 [D10C0100]          <1>             dd sret ;hd3  ; 8
000114D4 [D10C0100]          <1>             dd sret ;lpt  ; 9
000114D8 [D10C0100]          <1>             dd ocvt ;tty0 ; 10
000114DC [D10C0100]          <1>             dd ocvt ;tty1 ; 11
000114E0 [D10C0100]          <1>             dd ocvt ;tty2 ; 12
000114E4 [D10C0100]          <1>             dd ocvt ;tty3 ; 13
000114E8 [D10C0100]          <1>             dd ocvt ;tty4 ; 14
000114EC [D10C0100]          <1>             dd ocvt ;tty5 ; 15
000114F0 [D10C0100]          <1>             dd ocvt ;tty6 ; 16
000114F4 [D10C0100]          <1>             dd ocvt ;tty7 ; 17
000114F8 [D10C0100]          <1>             dd ocvt ;tty8 ; 18
000114FC [D10C0100]          <1>             dd ocvt ;tty9 ; 19
                             <1>             ;dd ocvt ;com1 ; 18
                             <1>             ;dd ocvt ;com2 ; 19
00011500 [D10C0100]          <1>             dd sret ;null ; 20
                             <1> KDEV_CADDR:
00011504 [D10C0100]          <1>             dd ctty ;tty  ; 1
00011508 [D10C0100]          <1>             dd cret ;mem  ; 2
0001150C [D10C0100]          <1>             dd cret ;fd0  ; 3
00011510 [D10C0100]          <1>             dd cret ;fd1  ; 4
00011514 [D10C0100]          <1>             dd cret ;hd0  ; 5
00011518 [D10C0100]          <1>             dd cret ;hd1  ; 6
0001151C [D10C0100]          <1>             dd cret ;hd2  ; 7
00011520 [D10C0100]          <1>             dd cret ;hd3  ; 8
00011524 [D10C0100]          <1>             dd cret ;lpt  ; 9
00011528 [D10C0100]          <1>             dd ocvt ;tty0 ; 10
0001152C [D10C0100]          <1>             dd ccvt ;tty1 ; 11
00011530 [D10C0100]          <1>             dd ccvt ;tty2 ; 12
00011534 [D10C0100]          <1>             dd ccvt ;tty3 ; 13
00011538 [D10C0100]          <1>             dd ccvt ;tty4 ; 14
0001153C [D10C0100]          <1>             dd ccvt ;tty5 ; 15
00011540 [D10C0100]          <1>             dd ccvt ;tty6 ; 16
00011544 [D10C0100]          <1>             dd ccvt ;tty7 ; 17
00011548 [D10C0100]          <1>             dd ccvt ;tty8 ; 18
0001154C [D10C0100]          <1>             dd ccvt ;tty9 ; 19
                             <1>             ;dd ccvt ;com1 ; 18
                             <1>             ;dd ccvt ;com2 ; 19
00011550 [D10C0100]          <1>             dd cret ;null ; 20
                             <1>
                             <1> KDEV_RADDR:
00011554 [D10C0100]          <1>             dd rtty ;tty  ; 1
00011558 [D10C0100]          <1>             dd rmem ;mem  ; 2
0001155C [D10C0100]          <1>             dd rfd  ;fd0  ; 3
00011560 [D10C0100]          <1>             dd rfd  ;fd1  ; 4
00011564 [D10C0100]          <1>             dd rhd  ;hd0  ; 5
00011568 [D10C0100]          <1>             dd rhd  ;hd1  ; 6
0001156C [D10C0100]          <1>             dd rhd  ;hd2  ; 7
00011570 [D10C0100]          <1>             dd rhd  ;hd3  ; 8
00011574 [D10C0100]          <1>             dd rlpt ;lpt  ; 9
00011578 [D10C0100]          <1>             dd rcvt ;tty0 ; 10
0001157C [D10C0100]          <1>             dd rcvt ;tty1 ; 11
00011580 [D10C0100]          <1>             dd rcvt ;tty2 ; 12
00011584 [D10C0100]          <1>             dd rcvt ;tty3 ; 13
00011588 [D10C0100]          <1>             dd rcvt ;tty4 ; 14
0001158C [D10C0100]          <1>             dd rcvt ;tty5 ; 15
00011590 [D10C0100]          <1>             dd rcvt ;tty6 ; 16
00011594 [D10C0100]          <1>             dd rcvt ;tty7 ; 17
00011598 [D10C0100]          <1>             dd rcvt ;tty8 ; 18
0001159C [D10C0100]          <1>             dd rcvt ;tty9 ; 19
                             <1>             ;dd rcvt ;com1 ; 18
                             <1>             ;dd rcvt ;com2 ; 19
000115A0 [C2000100]          <1>             dd rnull ;null ; 20
                             <1> KDEV_WADDR:
000115A4 [D10C0100]          <1>             dd wtty ;tty  ; 1
000115A8 [D10C0100]          <1>             dd wmem ;mem  ; 2
000115AC [D10C0100]          <1>             dd wfd  ;fd0  ; 3
000115B0 [D10C0100]          <1>             dd wfd  ;fd1  ; 4
000115B4 [D10C0100]          <1>             dd whd  ;hd0  ; 5
000115B8 [D10C0100]          <1>             dd whd  ;hd1  ; 6
000115BC [D10C0100]          <1>             dd whd  ;hd2  ; 7
000115C0 [D10C0100]          <1>             dd whd  ;hd3  ; 8
000115C4 [D10C0100]          <1>             dd wlpt ;lpt  ; 9
000115C8 [D10C0100]          <1>             dd xmtt ;tty0 ; 10
000115CC [D10C0100]          <1>             dd xmtt ;tty1 ; 11
000115D0 [D10C0100]          <1>             dd xmtt ;tty2 ; 12
000115D4 [D10C0100]          <1>             dd xmtt ;tty3 ; 13
000115D8 [D10C0100]          <1>             dd xmtt ;tty4 ; 14
```

```
 558 000115DC [D10C0100]          <1>             dd xmtt ;tty5 ; 15
 559 000115E0 [D10C0100]          <1>             dd xmtt ;tty6 ; 16
 560 000115E4 [D10C0100]          <1>             dd xmtt ;tty7 ; 17
 561 000115E8 [D10C0100]          <1>             dd xmtt ;tty8 ; 18
 562 000115EC [D10C0100]          <1>             dd xmtt ;tty9 ; 19
 563                              <1>             ;dd xmtt ;com1 ; 18
 564                              <1>             ;dd xmtt ;com2 ; 19
 565 000115F0 [C3000100]          <1>             dd wnull ;null ; 20
 566                              <1>
 567                              <1> ; DEV_ACCESS bits:
 568                              <1>       ; bit 0 = accessable by normal users
 569                              <1>       ; bit 1 = read access permission
 570                              <1>       ; bit 2 = write access permission
 571                              <1>       ; bit 3 = IOCTL permission to users
 572                              <1>       ; bit 4 = block device if it is set
 573                              <1>       ; bit 5 = 16 bit or 1024 byte data
 574                              <1>       ; bit 6 = 32 bit or 2048 byte data
 575                              <1>       ; bit 7 = installable device driver
 576                              <1>
 577                              <1> KDEV_ACCESS: ; 08/10/2016
 578 000115F4 07                 <1>             db  00000111b; tty, 1
 579 000115F5 07                 <1>             db  00000111b; mem, 2
 580 000115F6 8F                 <1>             db  10001111b; fd0, 3
 581 000115F7 8F                 <1>             db  10001111b; fd1, 4
 582 000115F8 8F                 <1>             db  10001111b; hd0, 5
 583 000115F9 8F                 <1>             db  10001111b; hd1, 6
 584 000115FA 8F                 <1>             db  10001111b; hd2, 7
 585 000115FB 8F                 <1>             db  10001111b; hd3, 8
 586 000115FC 07                 <1>             db  00000111b   ; lpt, 9
 587 000115FD 07                 <1>             db  00000111b; tty0, 10
 588 000115FE 07                 <1>             db  00000111b; tty1, 11
 589 000115FF 07                 <1>             db  00000111b; tty2, 12
 590 00011600 07                 <1>             db  00000111b; tty3, 13
 591 00011601 07                 <1>             db  00000111b; tty4, 14
 592 00011602 07                 <1>             db  00000111b; tty5, 15
 593 00011603 07                 <1>             db  00000111b; tty6, 16
 594 00011604 07                 <1>             db  00000111b; tty7, 17
 595 00011605 07                 <1>             db  00000111b; tty8, 18
 596 00011606 07                 <1>             db  00000111b; tty9, 19
 597                              <1>             ;db 00000111b; com1, 18
 598                              <1>             ;db 00000111b; com2, 19
 599 00011607 00                 <1>             db  00000000b   ; null, 0
 600                              <1>
 601                              <1> ; 07/10/2016
 602                              <1> NumOfInstallableDevices equ 8
 603                              <1> NUMIDEV           equ NumOfInstallableDevices ; 8
 604                              <1> NUMOFDEVICES equ NumOfKernelDevices + NumOfInstallableDevices
 605                              <1>
 606                              <1> ; 26/02/2017
 607                              <1> ; IRQ Callback (& Signal Response Byte) service availibity
 608                              <1> ; 'syscalbac'
 609                              <1> ; ***************************************************
 610                              <1> ; IRQ 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
 611                              <1> ; --- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 612                              <1> ; --- 00 00 00 01 02 03 00 04 00 05 06 07 08 09 00 00
 613                              <1> ; ***************************************************
 614                              <1> IRQenum:
 615 00011608 000000010203000400- <1>       db  0,0,0,1,2,3,0,4,0,5,6,7,8,9,0,0
 615 00011611 05060708090000      <1>
 616                              <1>
 617                              <1> ; 28/08/2017
 618                              <1> ; 20/08/2017
 619                              <1> ; DMA Registers (for 'sysdma')
 620                              <1> ; 02/07/2017 (sb16mod.s)
 621 00011618 00020406C0C4C8CC    <1> dma_adr:     db 0,2,4,6,0C0h,0C4h,0C8h,0CCh
 622 00011620 01030507C2C6CACE    <1> dma_cnt:     db 1,3,5,7,0C2h,0C6h,0CAh,0CEh
 623 00011628 878381828F8B898A    <1> dma_page:    db 87h,83h,81h,82h,8Fh,8Bh,89h,8Ah ; 03/08/2017
 624 00011630 0A0A0A0AD4D4D4D4    <1> dma_mask:    db 0Ah,0Ah,0Ah,0Ah,0D4h,0D4h,0D4h,0D4h
 625 00011638 0B0B0B0BD6D6D6D6    <1> dma_mod:     db 0Bh,0Bh,0Bh,0Bh,0D6h,0D6h,0D6h,0D6h
 626 00011640 0C0C0C0CD8D8D8D8    <1> dma_flip:    db 0Ch,0Ch,0Ch,0Ch,0D8h,0D8h,0D8h,0D8h
2314
2315                                  ; 27/08/2014
2316                                  scr_row:
2317 00011648 E0810B00                      dd 0B8000h + 0A0h + 0A0h + 0A0h ; Row 3
2318                                  scr_col:
2319 0001164C 00000000                      dd 0
2320
2321                                  Align 4
2322                                        ; 15/04/2016
2323                                        ; TRDOS 386 (TRDOS v2.0)
2324
2325                                        ; 21/08/2014
2326                                  ilist:
2327                                        ;times       32 dd cpu_except ; INT 0 to INT 1Fh
2328                                        ;
2329                                        ; Exception list
2330                                        ; 25/08/2014
2331 00011650 [17090000]                    dd     exc0   ; 0h,  Divide-by-zero Error
2332 00011654 [1E090000]                    dd     exc1
2333 00011658 [25090000]                    dd     exc2
2334 0001165C [2C090000]                    dd     exc3
2335 00011660 [30090000]                    dd     exc4
2336 00011664 [34090000]                    dd     exc5
2337 00011668 [38090000]                    dd     exc6   ; 06h,  Invalid Opcode
2338 0001166C [3C090000]                    dd     exc7
2339 00011670 [40090000]                    dd     exc8
2340 00011674 [44090000]                    dd     exc9
2341 00011678 [48090000]                    dd     exc10
2342 0001167C [4C090000]                    dd     exc11
2343 00011680 [50090000]                    dd     exc12
2344 00011684 [54090000]                    dd     exc13  ; 0Dh, General Protection Fault
2345 00011688 [58090000]                    dd     exc14  ; 0Eh, Page Fault
2346 0001168C [5C090000]                    dd     exc15
```

```
00011690 [60090000]                    dd     exc16
00011694 [64090000]                    dd     exc17
00011698 [68090000]                    dd     exc18
0001169C [6C090000]                    dd     exc19
000116A0 [70090000]                    dd     exc20
000116A4 [74090000]                    dd     exc21
000116A8 [78090000]                    dd     exc22
000116AC [7C090000]                    dd     exc23
000116B0 [80090000]                    dd     exc24
000116B4 [84090000]                    dd     exc25
000116B8 [88090000]                    dd     exc26
000116BC [8C090000]                    dd     exc27
000116C0 [90090000]                    dd     exc28
000116C4 [94090000]                    dd     exc29
000116C8 [98090000]                    dd     exc30
000116CC [9C090000]                    dd     exc31
                                 IRQ_list: ; 28/02/2017 ('syscalbac')
                                       ; Interrupt list
000116D0 [8B060000]                    dd     timer_int    ; INT 20h
                                              ;dd    irq0
000116D4 [FF0D0000]                    dd     kb_int       ; 24/01/2016
                                              ;dd    irq1
000116D8 [6D080000]                    dd     irq2
                                              ; COM2 int
000116DC [71080000]                    dd     irq3
                                              ; COM1 int
000116E0 [7C080000]                    dd     irq4
000116E4 [87080000]                    dd     irq5
                                 ;DISKETTE_INT: ;06/02/2015
000116E8 [B0410000]                    dd     fdc_int            ; 16/02/2015, IRQ 6 handler
                                              ;dd    irq6
                                 ; Default IRQ 7 handler against spurious IRQs (from master PIC)
                                 ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
000116EC [F60B0000]                    dd     default_irq7 ; 25/02/2015
                                              ;dd    irq7
                                 ; Real Time Clock Interrupt
000116F0 [F6070000]                    dd     rtc_int            ; 23/02/2015, IRQ 8 handler
                                              ;dd    irq8   ; INT 28h
000116F4 [97080000]                    dd     irq9
000116F8 [9B080000]                    dd     irq10
000116FC [9F080000]                    dd     irq11
00011700 [A3080000]                    dd     irq12
00011704 [A7080000]                    dd     irq13
                                 ;HDISK_INT1:  ;06/02/2015
00011708 [2D4B0000]                    dd     hdc1_int     ; 21/02/2015, IRQ 14 handler
                                              ;dd    irq14
                                 ;HDISK_INT2:  ;06/02/2015
0001170C [544B0000]                    dd     hdc2_int     ; 21/02/2015, IRQ 15 handler
                                              ;dd    irq15  ; INT 2Fh
                                              ; 14/08/2015
                                       ;dd    sysent       ; INT 30h (system calls)

                                       ; 15/04/2016
                                       ; TRDOS 386(TRDOS v2.0) Software Interrupts

00011710 [6D170100]                    dd     int30h       ; Reserved for
                                                           ; !!! Retro UNIX (RUNIX) !!!
                                                           ; !!! SINGLIX !!! System Calls
00011714 [F2140000]                    dd     int31h       ; Video BIOS (IBM PC/AT, Int 10h)
00011718 [1E0C0000]                    dd     int32h       ; Keyboard Functions (IBM PC/AT, Int 16h)
0001171C [66420000]                    dd     int33h       ; DISK I/O (IBM PC/AT, Int 13h)
00011720 [73F90000]                    dd     int34h       ; #IOCTL# (I/O port access support for ring 3)
00011724 [82590000]                    dd     int35h       ; Time/Date Functions (IBM PC/AT, Int 1Ah)
00011728 [AA0A0000]                    dd     ignore_int   ; INT 36h : Timer Functions
0001172C [AA0A0000]                    dd     ignore_int   ; INT 37h
00011730 [AA0A0000]                    dd     ignore_int   ; INT 38h
00011734 [AA0A0000]                    dd     ignore_int   ; INT 39h
00011738 [AA0A0000]                    dd     ignore_int   ; INT 3Ah
0001173C [AA0A0000]                    dd     ignore_int   ; INT 3Bh
00011740 [AA0A0000]                    dd     ignore_int   ; INT 3Ch
00011744 [AA0A0000]                    dd     ignore_int   ; INT 3Dh
00011748 [AA0A0000]                    dd     ignore_int   ; INT 3Eh
0001174C [AA0A0000]                    dd     ignore_int   ; INT 3Fh
00011750 [8CC50000]                    dd     sysent       ; INT 40h : !!! TRDOS 386 System Calls !!!
                                       ;dd    ignore_int
00011754 00000000                      dd     0

                                 ; 20/08/2014
                                   ; /* This is the default interrupt "handler" :-) */
                                   ; Linux v0.12 (head.s)
                                 int_msg:
00011758 556E6B6E6F776E2069-           db "Unknown interrupt ! ", 0
00011761 6E7465727275707420-
0001176A 212000

                                 ; 15/04/2016
                                 ; TRDOS 386 (TRDOS v2.0)

                                 ; 29/04/2016
                                 int30h:
                                 trdos_isc_routine:
                                       ; 02/05/2016
                                       ; 01/05/2016
                                       ; 29/04/2016
                                       ; 18/04/2016
                                       ; 15/04/2016 (TRDOS 386 = TRDOS v2.0)
                                       ; 17/04/2011 (TRDOS v1.0, 'IFC.ASM')
                                       ; 03/02/2011 ('trdos_ifc_routine')
                                       ;
0001176D 8B1C24                        mov    ebx, [esp] ; EIP (next)
00011770 83EB02                        sub    ebx, 2 ; EIP (CD ##h)

00011773 89C1                          mov    ecx, eax
```

```
00011775 8A4301                         mov    al, [ebx+1] ; CDh ##h

00011778 66BA1000                       mov    dx, KDATA
0001177C 8EDA                           mov    ds, dx
0001177E 8EC2                           mov    es, dx

00011780 FC                             cld
00011781 8B15[38580100]                 mov    edx, [k_page_dir]
00011787 0F22DA                         mov    cr3, edx

0001178A E83A1BFFFF                     call   bytetohex
0001178F 66A3[C5130100]                 mov    [int_num_str], ax

00011795 89D8                           mov    eax, ebx ; EIP
00011797 E86D1BFFFF                     call   dwordtohex
0001179C 8915[E1130100]                 mov    [eip_str], edx
000117A2 A3[E5130100]                   mov    [eip_str+4], eax

000117A7 89C8                           mov    eax, ecx
000117A9 E85B1BFFFF                     call   dwordtohex
000117AE 8915[D0130100]                 mov    [eax_str], edx
000117B4 A3[D4130100]                   mov    [eax_str+4], eax

000117B9 43                             inc    ebx
000117BA 8A03                           mov    al, [ebx] ; Interrupt number

                                  trdos_isc_handler:
000117BC 80FE30                         cmp    dh, 30h ; Retro UNIX, SINGLIX System calls
000117BF 7507                           jne    short trdos_usi_handler
000117C1 BE[62130100]                   mov    esi, isc_msg
000117C6 EB05                           jmp    short loc_write_inv_system_call_msg

                                  trdos_usi_handler:
000117C8 BE[78130100]                   mov    esi, usi_msg

                                  loc_write_inv_system_call_msg:
000117CD E88B4BFFFF                     call   print_msg
                                        ; 29/04/2016
000117D2 BE[AE130100]                   mov    esi, inv_msg_for_trdos_v2
000117D7 E8814BFFFF                     call   print_msg

                                  loc_ifc_terminate_process:
                                        ; u.uno = process number
                                        ; 29/04/2016

                                        ; 02/05/2016
000117DC FE05[5B030300]                 inc    byte [sysflg] ; 0FFh -> 0

000117E2 B801000000                     mov    eax, 1
000117E7 E979B0FFFF                     jmp    sysexit

                                  ; 07/03/2015
                                  ; Temporary Code
                                  display_disks:
000117EC 803D[F65C0000]00               cmp    byte [fd0_type], 0
000117F3 7605                           jna    short ddsks1
000117F5 E87D000000                     call   pdskm
                                  ddsks1:
000117FA 803D[F75C0000]00               cmp    byte [fd1_type], 0
00011801 760C                           jna    short ddsks2
00011803 C605[47190100]31               mov    byte [dskx], '1'
0001180A E868000000                     call   pdskm
                                  ddsks2:
0001180F 803D[F85C0000]00               cmp    byte [hd0_type], 0
00011816 7654                           jna    short ddsk6
00011818 66C705[45190100]68-            mov    word [dsktype], 'hd'
00011820 64
00011821 C605[47190100]30               mov    byte [dskx], '0'
00011828 E84A000000                     call   pdskm
                                  ddsks3:
0001182D 803D[F95C0000]00               cmp    byte [hd1_type], 0
00011834 7636                           jna    short ddsk6
00011836 C605[47190100]31               mov    byte [dskx], '1'
0001183D E835000000                     call   pdskm
                                  ddsks4:
00011842 803D[FA5C0000]00               cmp    byte [hd2_type], 0
00011849 7621                           jna    short ddsk6
0001184B C605[47190100]32               mov    byte [dskx], '2'
00011852 E820000000                     call   pdskm
                                  ddsks5:
00011857 803D[FB5C0000]00               cmp    byte [hd3_type], 0
0001185E 760C                           jna    short ddsk6
00011860 C605[47190100]33               mov    byte [dskx], '3'
00011867 E80B000000                     call   pdskm
                                  ddsk6:
0001186C BE[6F190100]                   mov    esi, nextline
00011871 E806000000                     call   pdskml
                                  pdskm_ok:
00011876 C3                             retn
                                  pdskm:
00011877 BE[43190100]                   mov    esi, dsk_ready_msg
                                  pdskml:
0001187C AC                             lodsb
0001187D 08C0                           or     al, al
0001187F 74F5                           jz     short pdskm_ok
00011881 56                             push   esi
                                        ; 13/05/2016
00011882 BB07000000                       mov      ebx, 7  ; Black background,
                                                     ; light gray forecolor
                                                     ; Video page 0 (bh=0)
00011887 E82604FFFF                     call   _write_tty
0001188C 5E                             pop    esi
0001188D EBED                           jmp    short pdskml
```

```

0001188F 90                       Align 2
                                        ; 21/08/2014
                                  exc_msg:
00011890 435055206578636570-            db "CPU exception ! "
00011899 74696F6E202120
                                  excnstr:           ; 25/08/2014
000118A0 3F3F68202045495020-            db "??h", "  EIP : "
000118A9 3A20
                                  EIPstr: ; 29/08/2014
000118AB 00<rept>                       times 12 db 0

                                        ; 23/02/2015
                                        ; 25/08/2014
                                  ;scounter:
                                  ;     db 5
                                  ;     db 19

                                  ; 06/11/2014
                                  ; Memory Information message
                                  ; 14/08/2015
                                  msg_memory_info:
000118B7 07                             db     07h
000118B8 0D0A                           db     0Dh, 0Ah
                                        ;db    "MEMORY ALLOCATION INFO", 0Dh, 0Ah, 0Dh, 0Ah
000118BA 546F74616C206D656D-            db     "Total memory : "
000118C3 6F7279203A20
                                  mem_total_b_str: ; 10 digits
000118C9 303030303030303030-            db     "0000000000 bytes", 0Dh, 0Ah
000118D2 302062797465730D0A
000118DB 202020202020202020-            db     "               ", 20h, 20h, 20h
000118E4 202020202020202020
                                  mem_total_p_str: ; 7 digits
000118ED 303030303030302070-            db     "0000000 pages", 0Dh, 0Ah
000118F6 616765730D0A
000118FC 0D0A                           db     0Dh, 0Ah
000118FE 46726565206D656D6F-            db     "Free memory  : "
00011907 727920203A20
                                  free_mem_b_str:  ; 10 digits
0001190D 3F3F3F3F3F3F3F3F3F-            db     "?????????? bytes", 0Dh, 0Ah
00011916 3F2062797465730D0A
0001191F 202020202020202020-            db     "               ", 20h, 20h, 20h
00011928 202020202020202020
                                  free_mem_p_str:  ; 7 digits
00011931 3F3F3F3F3F3F3F2070-            db     "??????? pages", 0Dh, 0Ah
0001193A 616765730D0A
00011940 0D0A00                         db     0Dh, 0Ah, 0

                                  dsk_ready_msg:
00011943 0D0A                           db     0Dh, 0Ah
                                  dsktype:
00011945 6664                           db     'fd'
                                  dskx:
00011947 30                             db     '0'
00011948 20                             db     20h
00011949 697320524541445920-            db     'is READY ...'
00011952 2E2E2E
00011955 00                             db     0

                                  setup_error_msg:
00011956 0D0A                           db 0Dh, 0Ah
00011958 4469736B2053657475-            db 'Disk Setup Error !'
00011961 70204572726F722021
0001196A 0D0A00                         db 0Dh, 0Ah,0

                                  next2line: ; 08/02/2016
0001196D 0D0A                           db     0Dh, 0Ah
                                  nextline:
0001196F 0D0A00                         db     0Dh, 0Ah, 0

                                  ; KERNEL - SYSINIT Messages
                                  ; 24/08/2015
                                  ; 13/04/2015 - (Retro UNIX 386 v1 Beginning)
                                  ; 14/07/2013
                                  ;kernel_init_err_msg:
                                  ;     db 0Dh, 0Ah
                                  ;     db 07h
                                  ;     db 'Kernel initialization ERROR !'
                                  ;     db 0Dh, 0Ah, 0

                                  ;welcome_msg:
                                  ;     db 0Dh, 0Ah
                                  ;     db 07h
                                  ;     db 'Welcome to TRDOS 386 Operating System !'
                                  ;     db 0Dh, 0Ah
                                  ;     db 'by Erdogan Tan - 31/12/2017 (v2.0.0)'
                                  ;     db 0Dh, 0Ah, 0

                                  panic_msg:
00011972 0D0A07                         db 0Dh, 0Ah, 07h
00011975 4552524F523A204B65-            db 'ERROR: Kernel Panic !'
0001197E 726E656C2050616E69-
00011987 632021
0001198A 0D0A00                         db 0Dh, 0Ah, 0

                                  ;msgl_drv_not_ready:
                                  ;     db 07h, 0Dh, 0Ah
                                  ;       db 'Drive not ready or read error !'
                                  ;       db 0Dh, 0Ah, 0

                                  starting_msg:
0001198D 5475726B6973682052-            db "Turkish Rational DOS v2.0 [31/12/2017] ...", 0
00011996 6174696F6E616C2044-
```

```
0001199F 4F532076322E30205B-
000119A8 33312F31322F323031-
000119B1 375D202E2E2E00
                                 NextLine:
000119B8 0D0A00                        db 0Dh, 0Ah, 0

                                 %include 'audio.s' ; 03/04/2017
                             <1> ; ****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - audio.s
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 28/10/2017
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 03/04/2017
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ****************************************************************************
                             <1> 
                             <1> ; AUDIO CONTROLLER & CODEC DEFINITIONS & CODE FOR TRDOS 386
                             <1> 
                             <1> ;=============================================================================
                             <1> ;               EQUATES
                             <1> ;=============================================================================
                             <1> 
                             <1> ; PCI EQUATES
                             <1> 
                             <1> BIT0  EQU 1
                             <1> BIT1  EQU 2
                             <1> BIT2  EQU 4
                             <1> BIT3  EQU 8
                             <1> BIT4  EQU 10h
                             <1> BIT5  EQU 20h
                             <1> BIT6  EQU 40h
                             <1> BIT7  EQU 80h
                             <1> BIT8  EQU 100h
                             <1> BIT9  EQU 200h
                             <1> BIT10 EQU 400h
                             <1> BIT11 EQU 800h
                             <1> BIT12 EQU 1000h
                             <1> BIT13 EQU 2000h
                             <1> BIT14 EQU 4000h
                             <1> BIT15 EQU 8000h
                             <1> BIT16 EQU 10000h
                             <1> BIT17 EQU 20000h
                             <1> BIT18 EQU 40000h
                             <1> BIT19 EQU 80000h
                             <1> BIT20 EQU 100000h
                             <1> BIT21 EQU 200000h
                             <1> BIT22 EQU 400000h
                             <1> BIT23 EQU 800000h
                             <1> BIT24 EQU 1000000h
                             <1> BIT25 EQU 2000000h
                             <1> BIT26 EQU 4000000h
                             <1> BIT27 EQU 8000000h
                             <1> BIT28 EQU 10000000h
                             <1> BIT29 EQU 20000000h
                             <1> BIT30 EQU 40000000h
                             <1> BIT31 EQU 80000000h
                             <1> NOT_BIT31 EQU 7FFFFFFFh
                             <1> 
                             <1> ; PCI equates
                             <1> ; PCI function address (PFA)
                             <1> ; bit 31 = 1
                             <1> ; bit 23:16 = bus number     (0-255)
                             <1> ; bit 15:11 = device number  (0-31)
                             <1> ; bit 10:8 = function number (0-7)
                             <1> ; bit 7:0 = register number  (0-255)
                             <1> 
                             <1> IO_ADDR_MASK    EQU     0FFFEh  ; mask off bit 0 for reading BARs
                             <1> PCI_INDEX_PORT  EQU     0CF8h
                             <1> PCI_DATA_PORT   EQU     0CFCh
                             <1> PCI32           EQU     BIT31   ; bitflag to signal 32bit access
                             <1> PCI16           EQU     BIT30   ; bitflag for 16bit access
                             <1> NOT_PCI32_PCI16    EQU    03FFFFFFFh ; NOT BIT31+BIT30 ; 19/03/2017
                             <1> 
                             <1> PCI_FN0         EQU     0 << 8
                             <1> PCI_FN1         EQU     1 << 8
                             <1> PCI_FN2         EQU     2 << 8
                             <1> PCI_FN3         EQU     3 << 8
                             <1> PCI_FN4         EQU     4 << 8
                             <1> PCI_FN5         EQU     5 << 8
                             <1> PCI_FN6         EQU     6 << 8
                             <1> PCI_FN7         EQU     7 << 8
                             <1> 
                             <1> PCI_CMD_REG  EQU    04h    ; reg 04, command reg
                             <1> IO_ENA             EQU    BIT0   ; i/o decode enable
                             <1> MEM_ENA            EQU    BIT1   ; memory decode enable
                             <1> BM_ENA             EQU     BIT2 ; bus master enable
                             <1> 
                             <1> ; VIA VT8233 EQUATES
                             <1> 
                             <1> VIA_VID            equ 1106h    ; VIA's PCI vendor ID
                             <1> VT8233_DID      equ 3059h ; VT8233 (VT8235) device ID
                             <1> 
                             <1> PCI_IO_BASE          equ 10h
                             <1> AC97_INT_LINE        equ 3Ch
                             <1> VIA_ACLINK_CTRL      equ 41h
                             <1> VIA_ACLINK_STAT      equ 40h
                             <1> VIA_ACLINK_C00_READY equ 01h ; primary codec ready
                             <1> 
                             <1> VIA_REG_AC97      equ 80h ; dword
                             <1> 
                             <1> VIA_ACLINK_CTRL_ENABLE    equ   80h ; 0: disable, 1: enable
                             <1> VIA_ACLINK_CTRL_RESET     equ   40h ; 0: assert, 1: de-assert
```

```
<1> VIA_ACLINK_CTRL_SYNC      equ   20h ; 0: release SYNC, 1: force SYNC hi
<1> VIA_ACLINK_CTRL_VRA       equ   08h ; 0: disable VRA, 1: enable VRA
<1> VIA_ACLINK_CTRL_PCM       equ   04h ; 0: disable PCM, 1: enable PCM
<1> VIA_ACLINK_CTRL_INIT      equ  (VIA_ACLINK_CTRL_ENABLE + VIA_ACLINK_CTRL_RESET + VIA_ACLINK_CTRL_PCM + VIA_ACLINK_CTRL_VRA)
<1>
<1> CODEC_AUX_VOL             equ   04h
<1> VIA_REG_AC97_BUSY  equ   01000000h ;(1<<24)
<1> VIA_REG_AC97_CMD_SHIFT    equ   10h ; 16
<1> VIA_REG_AC97_PRIMARY_VALID equ 02000000h ;(1<<25)
<1> VIA_REG_AC97_READ  equ   00800000h ;(1<<23)
<1> VIA_REG_AC97_CODEC_ID_SHIFT   equ  1Eh ; 30
<1> VIA_REG_AC97_CODEC_ID_PRIMARY equ  0
<1> VIA_REG_AC97_DATA_SHIFT equ   0
<1> VIADEV_PLAYBACK         equ   0
<1> VIA_REG_OFFSET_STATUS   equ   0    ;; byte - channel status
<1> VIA_REG_OFFSET_CONTROL  equ   01h  ;; byte - channel control
<1> VIA_REG_CTRL_START equ   80h  ;; WO
<1> VIA_REG_CTRL_TERMINATE  equ   40h  ;; WO
<1> VIA_REG_CTRL_PAUSE      equ   08h  ;; RW
<1> VIA_REG_CTRL_RESET      equ   01h  ;; RW - probably reset? undocumented
<1> VIA_REG_OFFSET_STOP_IDX equ   08h  ;; dword - stop index, channel type, sample rate
<1> VIA8233_REG_TYPE_16BIT  equ   200000h ;; RW
<1> VIA8233_REG_TYPE_STEREO equ   100000h ;; RW
<1> VIA_REG_OFFSET_CURR_INDEX equ 0Fh ;; byte - channel current index (for via8233 only)
<1> VIA_REG_OFFSET_TABLE_PTR equ  04h  ;; dword - channel table pointer
<1> VIA_REG_OFFSET_CURR_PTR equ   04h  ;; dword - channel current pointer
<1> VIA_REG_OFS_PLAYBACK_VOLUME_L equ  02h ;; byte
<1> VIA_REG_OFS_PLAYBACK_VOLUME_R equ  03h ;; byte
<1> VIA_REG_CTRL_AUTOSTART    equ   20h
<1> VIA_REG_CTRL_INT_EOL      equ   02h
<1> VIA_REG_CTRL_INT_FLAG     equ   01h
<1> VIA_REG_CTRL_INT   equ  (VIA_REG_CTRL_INT_FLAG + VIA_REG_CTRL_INT_EOL + VIA_REG_CTRL_AUTOSTART)
<1>
<1> VIA_REG_STAT_STOPPED      equ   04h    ;; RWC
<1> VIA_REG_STAT_EOL   equ   02h    ;; RWC
<1> VIA_REG_STAT_FLAG  equ   01h    ;; RWC
<1> VIA_REG_STAT_ACTIVE       equ   80h    ;; RO
<1> ; 28/11/2016
<1> VIA_REG_STAT_LAST  equ   40h    ;; RO
<1> VIA_REG_STAT_TRIGGER_QUEUED equ 08h  ;; RO
<1> VIA_REF_CTRL_INT_STOP     equ   04h  ; Interrupt on Current Index = Stop Index
<1>                               ; and End of Block
<1>
<1> VIA_REG_OFFSET_CURR_COUNT equ 0Ch ;; dword - channel current count, index
<1>
<1> PORTB        EQU    061h
<1> REFRESH_STATUS     EQU    010h   ; Refresh signal status
<1>
<1> ; AC97 Codec registers.
<1>
<1> ; each codec/mixer register is 16bits
<1>
<1> CODEC_RESET_REG                 equ     00h  ; reset codec
<1> CODEC_MASTER_VOL_REG            equ     02h  ; master volume
<1> CODEC_HP_VOL_REG                equ     04h  ; headphone volume
<1> CODEC_MASTER_MONO_VOL_REG       equ     06h  ; master mono volume
<1> CODEC_MASTER_TONE_REG           equ     08h  ; master tone (R+L)
<1> CODEC_PCBEEP_VOL_REG            equ     0Ah  ; PC beep volume
<1> CODEC_PHONE_VOL_REG             equ     0Bh  ; phone volume
<1> CODEC_MIC_VOL_REG               equ     0Eh  ; MIC volume
<1> CODEC_LINE_IN_VOL_REG           equ     10h  ; line input volume
<1> CODEC_CD_VOL_REG                equ     12h  ; CD volume
<1> CODEC_VID_VOL_REG               equ     14h  ; video volume
<1> CODEC_AUX_VOL_REG               equ     16h  ; aux volume
<1> CODEC_PCM_OUT_REG               equ     18h  ; PCM output volume
<1> CODEC_RECORD_SELECT_REG         equ     1Ah  ; record select input
<1> CODEC_RECORD_VOL_REG            equ     1Ch  ; record volume
<1> CODEC_RECORD_MIC_VOL_REG        equ     1Eh  ; record mic volume
<1> CODEC_GP_REG                    equ     20h  ; general purpose
<1> CODEC_3D_CONTROL_REG            equ     22h  ; 3D control
<1> ; 24h is reserved
<1> CODEC_POWER_CTRL_REG            equ     26h  ; powerdown control
<1> CODEC_EXT_AUDIO_REG             equ     28h  ; extended audio
<1> CODEC_EXT_AUDIO_CTRL_REG        equ     2Ah  ; extended audio control
<1> CODEC_PCM_FRONT_DACRATE_REG     equ     2Ch  ; PCM out sample rate
<1> CODEC_PCM_SURND_DACRATE_REG     equ     2Eh  ; surround sound sample rate
<1> CODEC_PCM_LFE_DACRATE_REG       equ     30h  ; LFE sample rate
<1> CODEC_LR_ADCRATE_REG            equ     32h  ; PCM in sample rate
<1> CODEC_MIC_ADCRATE_REG           equ     34h  ; mic in sample rate
<1>
<1> ; VT8233 SGD bits (21/04/2017)
<1> FLAG  EQU BIT30
<1> EOL   EQU BIT31
<1>
<1> ; INTEL ICH EQUATES
<1> ; 28/05/2017
<1> INTEL_VID    equ    8086h  ; Intel's PCI vendor ID
<1> ICH_DID             equ    2415h  ; ICH (82801AA) device ID
<1> NAMBAR_REG       equ       10h    ; native audio mixer Base Address Register
<1> NABMBAR_REG      equ       14h    ; native audio bus mastering Base Addr Reg
<1>
<1> PI_CR_REG        equ       0Bh      ; PCM in Control Register
<1> PO_CR_REG    equ      1Bh      ; PCM out Control Register
<1> MC_CR_REG    equ      2Bh      ; MIC in Control Register
<1>
<1> PI_SR_REG    equ      6        ; PCM in Status register
<1> PO_SR_REG    equ      16h      ; PCM out Status register
<1> MC_SR_REG    equ      26h      ; MIC in Status register
<1>
<1> IOCE         equ      BIT4     ; interrupt on complete enable.
<1> FEIFE        equ      BIT3     ; set if you want an interrupt to fire
```

```
                             <1> LVBIE       equ     BIT2    ; last valid buffer interrupt enable.
                             <1> RR          equ     BIT1    ; reset registers.  Nukes all regs
                             <1>                                 ; except bits 4:2 of this register.
                             <1>                                 ; Only set this bit if BIT 0 is 0
                             <1> RPBM        equ     BIT0    ; Run/Pause
                             <1>                           ; set this bit to start the codec!
                             <1> 
                             <1> PI_BDBAR_REG equ    0       ; PCM in buffer descriptor BAR
                             <1> PO_BDBAR_REG equ    10h     ; PCM out buffer descriptor BAR
                             <1> MC_BDBAR_REG equ    20h     ; MIC in buffer descriptor BAR
                             <1> 
                             <1> PI_CIV_REG  equ     4       ; PCM in current Index value (RO)
                             <1> PO_CIV_REG  equ     14h     ; PCM out current Index value (RO)
                             <1> MC_CIV_REG  equ     24h     ; MIC in current Index value (RO)
                             <1> 
                             <1> PI_LVI_REG  equ     5       ; PCM in Last Valid Index
                             <1> PO_LVI_REG  equ     15h     ; PCM out Last Valid Index
                             <1> MC_LVI_REG  equ     25h     ; MIC in Last Valid Index
                             <1> 
                             <1> IOC         equ     BIT31; Fire an interrupt whenever this
                             <1>                           ; buffer is complete.
                             <1> BUP         equ     BIT30; Buffer Underrun Policy.
                             <1> 
                             <1> GLOB_CNT_REG equ    2Ch     ; Global Control Register
                             <1> GLOB_STS_REG equ    30h     ; Global Status register (RO)
                             <1> 
                             <1> CTRL_ST_CREADY     equ   BIT8+BIT9+BIT28 ; Primary Codec Ready
                             <1> 
                             <1> CODEC_REG_POWERDOWN   equ 26h
                             <1> CODEC_REG_ST          equ 26h
                             <1> 
                             <1> ; 22/06/2017
                             <1> PO_PICB_REG  equ 18h      ; PCM Out Position In Current Buffer Register
                             <1> 
                             <1> ;=============================================================================
                             <1> ;               CODE
                             <1> ;=============================================================================
                             <1> 
                             <1> ; CODE for INTEL ICH AC'97 AUDIO CONTROLLER
                             <1> 
                             <1> DetectICH:
                             <1>       ; 10/06/2017
                             <1>       ; 05/06/2017
                             <1>       ; 29/05/2017
                             <1>       ; 28/05/2017
000119BB B886801524          <1>       mov     eax, (ICH_DID << 16) + INTEL_VID
000119C0 E876000000          <1>         call    pciFindDevice
000119C5 730D                <1>         jnc     short d_ac97_1
                             <1> d_ac97_0:
                             <1> ; couldn't find the audio device!
000119C7 C3                  <1>       retn
                             <1> 
                             <1> ; CODE for VIA VT8233 AUDIO CONTROLLER
                             <1> 
                             <1> DetectVT8233:
                             <1>       ; 10/06/2017
                             <1>       ; 05/06/2017
                             <1>       ; 29/05/2017
                             <1>       ; 03/04/2017
000119C8 B806115930          <1>       mov     eax, (VT8233_DID << 16) + VIA_VID
000119CD E869000000          <1>         call    pciFindDevice
                             <1> ;       jnc     short d_vt8233_0
                             <1> ; couldn't find the audio device!
                             <1> ;     retn
000119D2 72F3                <1>       jc     short d_ac97_0  ; 28/05/2017
                             <1> d_vt8233_0:
                             <1>       ; 24/03/2017 ('player.asm')
                             <1>       ; 12/11/2016
                             <1>       ; Erdogan Tan - 8/11/2016
                             <1>       ; References: Kolibrios - vt823x.asm (2016)
                             <1>       ;            VIA VT8235 V-Link South Bridge (VT8235-VIA.PDF)(2002)
                             <1>       ;            lowlevel.eu - AC97 (2016)
                             <1>       ;            .wav player for DOS by Jeff Leyda (2002) -this file-
                             <1>       ;            Linux kernel - via82xx.c (2016)
                             <1> d_ac97_1:
                             <1>       ; eax = BUS/DEV/FN
                             <1>       ;      00000000BBBBBBBBDDDDDFFF00000000
                             <1>       ; edx = DEV/VENDOR
                             <1>       ;      DDDDDDDDDDDDDDDDVVVVVVVVVVVVVVVV
                             <1> 
000119D4 A3[C46B0100]        <1>       mov    [audio_dev_id], eax
000119D9 8915[C86B0100]      <1>       mov    [audio_vendor], edx
                             <1> 
                             <1>       ; init controller
000119DF B004                <1>       mov    al, PCI_CMD_REG ; command register (04h)
000119E1 E8E2000000          <1>       call   pciRegRead32
                             <1> 
                             <1>       ; eax = BUS/DEV/FN/REG
                             <1>       ; edx = STATUS/COMMAND
                             <1>       ;      SSSSSSSSSSSSSSSSCCCCCCCCCCCCCCCC
000119E6 8915[CC6B0100]      <1>       mov    [audio_stats_cmd], edx
                             <1> 
000119EC B010                <1>       mov    al, PCI_IO_BASE ; IO base address register (10h)
                             <1>       ;mov   al, NAMBAR_REG      ; Native Audio Mixer BAR (10h)
000119EE E8D5000000          <1>       call   pciRegRead32
                             <1> 
000119F3 66813D[C86B0100]86- <1>       cmp    word [audio_vendor], 8086h ; AC'97 ?
000119FB 80                  <1>
000119FC 751F                <1>       jne    short d_vt8233_1
                             <1> 
000119FE 6683E2FE            <1>       and    dx, 0FFFEh ; Audio Codec IO_ADDR_MASK
00011A02 668915[F46B0100]    <1>       mov    [NAMBAR], dx
                             <1> 
```

```
00011A09 B014                    <1>       mov    al, NABMBAR_REG ; Native Audio Bus Mastering BAR (14h)
00011A0B E8B8000000              <1>       call   pciRegRead32
                                 <1> 
00011A10 6683E2C0                <1>       and    dx, 0FFC0h ; Audio Controller IO_ADDR_MASK
00011A14 668915[F66B0100]        <1>       mov    [NABMBAR], dx
                                 <1>         ;mov [audio_io_base], dx
                                 <1> 
00011A1B EB0B                    <1>       jmp    short d_ac97_2
                                 <1> 
                                 <1> d_vt8233_1:
00011A1D 6683E2C0                <1>       and     dx, 0FFC0h ; Audio Controller IO_ADDR_MASK
00011A21 668915[C26B0100]        <1>         mov     [audio_io_base], dx
                                 <1> 
                                 <1> d_ac97_2:
                                 <1>       ; 10/06/2017
00011A28 B03C                    <1>       mov    al, AC97_INT_LINE ; Interrupt Line Register (3Ch)
                                 <1>       ;call  pciRegRead32
00011A2A E886000000              <1>       call   pciRegRead8
                                 <1> 
                                 <1>       ;and   edx, 0FFh
00011A2F 6681E2FF00              <1>       and    dx, 0FFh
                                 <1> 
00011A34 8815[BF6B0100]          <1>       mov     [audio_intr], dl
                                 <1> 
00011A3A C3                      <1>       retn
                                 <1> 
                                 <1>       ;; (Note: Interrupts are already enabled by TRDOS 386 kernel!)
                                 <1>       ;mov   cx, dx
                                 <1> 
                                 <1>       ;in    al, 0A1h ; irq 8-15
                                 <1>       ;mov   ah, al
                                 <1>       ;in    al, 21h  ; irq 0-7
                                 <1>       ;btr   ax, dx  ; unmask ; 17/03/2017
                                 <1>       ;;bts  ax, dx   ; MASK interrupt ; 10/06/2017
                                 <1>       ;out   21h, al  ; irq <= 7
                                 <1>       ;mov   al, ah
                                 <1>       ;out   0A1h, al ; irq > 7
                                 <1>       ;
                                 <1> 
                                 <1>       ; 10/06/2017
                                 <1>       ; === Intel ICH I/O Controller Hub Datasheet, Section 8.1.16 ===
                                 <1>       ; PRQ[n]_ROUT Register (61h, PRQB) Bit 7:
                                 <1>       ; Interrupt Routing Enable (IRQEN).
                                 <1>       ; 0 = The corresponding PIRQ is routed to one of the ISA-compatible
                                 <1>       ;     interrupts specified in bits[3:0].
                                 <1>       ; 1 = The PIRQ is not routed to the 8259.
                                 <1>       ; Note: If the PIRQ is intended to cause an interrupt to the ICH’s
                                 <1>       ;      integrated I/O APIC, then this bit should be set to 0 and
                                 <1>       ;      the APIC_EN bit should be set to 1.
                                 <1>       ;      The IRQEN must be set to 0 and the PIRQ routed to
                                 <1>       ;      an 8259 interrupt via the IRQ Routing filed (bits[3:0).
                                 <1>       ;      The corresponding 8259 interrupt must be masked via the
                                 <1>       ;      appropriated bit in the 8259’s OCW1 (Interrupt Mask)
                                 <1>       ;      register. The IOAPIC must then be enabled by setting
                                 <1>       ;      the APIC_EN bit in the GEN_CNTL register.
                                 <1> 
                                 <1>       ;mov   eax, 0F861h  ; D31:F0
                                 <1>              ;AL=61h : PIRQ[B] Routing Control Reg, LPC interface
                                 <1>       ;;mov  dl,  [audio_intr]
                                 <1>       ;call  pciRegWrite8
                                 <1>       ;;mov  al, 0D0h ; General Control Register (GEN_CTL)
                                 <1>       ;;call pciRegRead32
                                 <1>       ;;or   edx, 100h ; Bit 8, APIC_EN (Enable I/O APIC)
                                 <1>       ;;;call      pciRegWrite32
                                 <1>       ;;and  edx, ~100h
                                 <1>       ;;call pciRegWrite32 ; ; Bit 8, APIC_EN (Disable I/O APIC)
                                 <1>       ;
                                 <1> 
                                 <1>       ;mov   dx, 4D1h     ; 8259 ELCR2
                                 <1>       ;in    al, dx
                                 <1>       ;mov   ah, al
                                 <1>       ;;mov  dx, 4D0h     ; 8259 ELCR1
                                 <1>       ;dec   dl
                                 <1>       ;in    al, dx
                                 <1>       ;bts   ax, cx
                                 <1>       ;;mov  dx, 4D0h
                                 <1>       ;out   dx, al       ; set level-triggered mode
                                 <1>       ;mov   al, ah ; 29/05/2017
                                 <1>       ;;mov  dx, 4D1h
                                 <1>       ;inc   dl
                                 <1>       ;out   dx, al       ; set level-triggered mode
                                 <1> 
                                 <1>       ;xor   eax, eax ; 0
                                 <1> 
                                 <1>       ;retn
                                 <1> 
                                 <1> ; CODE for PCI
                                 <1> 
                                 <1> pciFindDevice:
                                 <1>       ; 03/04/2017 ('pci.asm', 20/03/2017)
                                 <1>       ;
                                 <1>       ; scan through PCI space looking for a device+vendor ID
                                 <1>       ;
                                 <1>       ; Entry: EAX=Device+Vendor ID
                                 <1>       ;
                                 <1>       ; Exit: EAX=PCI address if device found
                                 <1>       ;       EDX=Device+Vendor ID
                                 <1>       ;         CY clear if found, set if not found. EAX invalid if CY set.
                                 <1>       ;
                                 <1>       ; Destroys: ebx, esi, edi, cl
                                 <1>       ;
                                 <1> 
                                 <1>       ;push  ecx
```

```
00011A3B 50                  <1>       push   eax
                             <1>       ;push  esi
                             <1>       ;push  edi
                             <1>
00011A3C 89C6                <1>         mov     esi, eax                ; save off vend+device ID
00011A3E BF00FFFF7F          <1>         mov     edi, (80000000h - 100h) ; start with bus 0, dev 0 func 0
                             <1>
                             <1> nextPCIdevice:
00011A43 81C700010000        <1>         add     edi, 100h
00011A49 81FF00F8FF80        <1>         cmp     edi, 80FFF800h         ; scanned all devices?
00011A4F F9                  <1>         stc
00011A50 740C                <1>         je      short PCIScanExit       ; not found
                             <1>
00011A52 89F8                <1>         mov     eax, edi                ; read PCI registers
00011A54 E86F000000          <1>         call    pciRegRead32
00011A59 39F2                <1>         cmp     edx, esi                ; found device?
00011A5B 75E6                <1>         jne     short nextPCIdevice
00011A5D F8                  <1>         clc
                             <1>
                             <1> PCIScanExit:
00011A5E 9C                  <1>       pushf
00011A5F B8FFFFFF7F          <1>       mov    eax, NOT_BIT31      ; 19/03/2017
00011A64 21F8                <1>       and    eax, edi     ; return only bus/dev/fn #
00011A66 9D                  <1>       popf
                             <1>
                             <1>       ;pop   edi
                             <1>       ;pop   esi
00011A67 5A                  <1>       pop    edx
                             <1>       ;pop   ecx
00011A68 C3                  <1>       retn
                             <1>
                             <1> pciRegRead:
                             <1>       ; 03/04/2017 ('pci.asm', 20/03/2017)
                             <1>       ;
                             <1>       ; 8/16/32bit PCI reader
                             <1>       ;
                             <1>       ; Entry: EAX=PCI Bus/Device/fn/register number
                             <1>       ;           BIT30 set if 32 bit access requested
                             <1>       ;           BIT29 set if 16 bit access requested
                             <1>       ;           otherwise defaults to 8 bit read
                             <1>       ;
                             <1>       ; Exit:  DL,DX,EDX register data depending on requested read size
                             <1>       ;
                             <1>       ; Note1: this routine is meant to be called via pciRegRead8,
                             <1>       ;       pciRegread16 or pciRegRead32, listed below.
                             <1>       ;
                             <1>       ; Note2: don't attempt to read 32 bits of data from a non dword
                             <1>       ;       aligned reg number.  Likewise, don't do 16 bit reads from
                             <1>       ;       non word aligned reg #
                             <1>
00011A69 53                  <1>       push   ebx
00011A6A 51                  <1>       push   ecx
00011A6B 89C3                <1>         mov     ebx, eax        ; save eax, dh
00011A6D 88F1                <1>         mov     cl, dh
                             <1>
00011A6F 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; clear out data size request
00011A74 0D00000080          <1>         or      eax, BIT31             ; make a PCI access request
00011A79 24FC                <1>         and     al, ~3 ; NOT 3         ; force index to be dword
                             <1>
00011A7B 66BAF80C            <1>         mov     dx, PCI_INDEX_PORT
00011A7F EF                  <1>         out dx, eax                    ; write PCI selector
                             <1>
00011A80 66BAFC0C            <1>         mov     dx, PCI_DATA_PORT
00011A84 88D8                <1>         mov     al, bl
00011A86 2403                <1>         and     al, 3                  ; figure out which port to
00011A88 00C2                <1>         add     dl, al                 ; read to
                             <1>
00011A8A F7C3000000C0        <1>       test    ebx, PCI32+PCI16
00011A90 7507                <1>         jnz     short _pregr0
00011A92 EC                  <1>       in     al, dx              ; return 8 bits of data
00011A93 88C2                <1>         mov dl, al
00011A95 88CE                <1>       mov     dh, cl                   ; restore dh for 8 bit read
00011A97 EB12                <1>       jmp    short _pregr2
                             <1> _pregr0:
00011A99 F7C300000080        <1>       test    ebx, PCI32
00011A9F 7507                <1>         jnz short _pregr1
00011AA1 66ED                <1>       in     ax, dx
00011AA3 6689C2              <1>         mov     dx, ax                 ; return 16 bits of data
00011AA6 EB03                <1>       jmp    short _pregr2
                             <1> _pregr1:
00011AA8 ED                  <1>       in     eax, dx                  ; return 32 bits of data
00011AA9 89C2                <1>       mov    edx, eax
                             <1> _pregr2:
00011AAB 89D8                <1>       mov     eax, ebx          ; restore eax
00011AAD 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; clear out data size request
00011AB2 59                  <1>       pop    ecx
00011AB3 5B                  <1>       pop    ebx
00011AB4 C3                  <1>       retn
                             <1>
                             <1> pciRegRead8:
00011AB5 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; set up 8 bit read size
00011ABA EBAD                <1>         jmp     short pciRegRead; call generic PCI access
                             <1>
                             <1> pciRegRead16:
00011ABC 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; set up 16 bit read size
00011AC1 0D00000040          <1>         or      eax, PCI16             ; call generic PCI access
00011AC6 EBA1                <1>         jmp     short pciRegRead
                             <1>
                             <1> pciRegRead32:
00011AC8 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; set up 32 bit read size
00011ACD 0D00000080          <1>         or      eax, PCI32             ; call generic PCI access
00011AD2 EB95                <1>         jmp     pciRegRead
                             <1>
                             <1> pciRegWrite:
                             <1>       ; 03/04/2017 ('pci.asm', 29/11/2016)
                             <1>       ;
                             <1>       ; 8/16/32bit PCI writer
                             <1>       ;
                             <1>       ; Entry: EAX=PCI Bus/Device/fn/register number
                             <1>       ;           BIT31 set if 32 bit access requested
                             <1>       ;           BIT30 set if 16 bit access requested
                             <1>       ;           otherwise defaults to 8bit read
                             <1>       ;        DL/DX/EDX data to write depending on size
                             <1>       ;
                             <1>       ; Note1: this routine is meant to be called via pciRegWrite8,
                             <1>       ;       pciRegWrite16 or pciRegWrite32 as detailed below.
                             <1>       ;
                             <1>       ; Note2: don't attempt to write 32bits of data from a non dword
                             <1>       ;       aligned reg number.  Likewise, don't do 16 bit writes from
                             <1>       ;       non word aligned reg #
                             <1>
00011AD4 53                  <1>       push   ebx
00011AD5 51                  <1>       push   ecx
00011AD6 89C3                <1>         mov     ebx, eax        ; save eax, edx
00011AD8 89D1                <1>         mov     ecx, edx
00011ADA 25FFFFFF3F          <1>       and     eax, NOT_PCI32_PCI16     ; clear out data size request
00011ADF 0D00000080          <1>         or      eax, BIT31             ; make a PCI access request
00011AE4 24FC                <1>         and     al, ~3 ; NOT 3         ; force index to be dword
                             <1>
00011AE6 66BAF80C            <1>         mov     dx, PCI_INDEX_PORT
00011AEA EF                  <1>         out dx, eax                    ; write PCI selector
                             <1>
00011AEB 66BAFC0C            <1>         mov     dx, PCI_DATA_PORT
00011AEF 88D8                <1>         mov     al, bl
00011AF1 2403                <1>         and     al, 3                  ; figure out which port to
00011AF3 00C2                <1>         add     dl, al                 ; write to
                             <1>
00011AF5 F7C3000000C0        <1>       test    ebx, PCI32+PCI16
00011AFB 7505                <1>         jnz     short _pregw0
00011AFD 88C8                <1>       mov    al, cl                    ; put data into al
00011AFF EE                  <1>       out    dx, al
00011B00 EB12                <1>       jmp    short _pregw2
                             <1> _pregw0:
00011B02 F7C300000080        <1>       test    ebx, PCI32
00011B08 7507                <1>         jnz     short _pregw1
00011B0A 6689C8              <1>       mov    ax, cx              ; put data into ax
00011B0D 66EF                <1>       out    dx, ax
00011B0F EB03                <1>       jmp    short _pregw2
                             <1> _pregw1:
00011B11 89C8                <1>       mov    eax, ecx            ; put data into eax
00011B13 EF                  <1>       out    dx, eax
                             <1> _pregw2:
00011B14 89D8                <1>         mov     eax, ebx        ; restore eax
00011B16 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; clear out data size request
00011B1B 89CA                <1>         mov     edx, ecx        ; restore dx
00011B1D 59                  <1>       pop    ecx
00011B1E 5B                  <1>       pop    ebx
00011B1F C3                  <1>       retn
                             <1>
                             <1> pciRegWrite8:
00011B20 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; set up 8 bit write size
00011B25 EBAD                <1>         jmp short pciRegWrite   ; call generic PCI access
                             <1>
                             <1> pciRegWrite16:
00011B27 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; set up 16 bit write size
00011B2C 0D00000040          <1>         or      eax, PCI16             ; call generic PCI access
00011B31 EBA1                <1>         jmp short pciRegWrite
                             <1>
                             <1> pciRegWrite32:
00011B33 25FFFFFF3F          <1>         and     eax, NOT_PCI32_PCI16   ; set up 32 bit write size
00011B38 0D00000080          <1>         or      eax, PCI32             ; call generic PCI access
00011B3D EB95                <1>         jmp pciRegWrite
                             <1>
                             <1> init_codec:
                             <1>       ; 05/06/2017
                             <1>       ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
                             <1>       ;
00011B3F A1[C46B0100]        <1>       mov    eax, [audio_dev_id]
00011B44 B041                <1>       mov    al, VIA_ACLINK_CTRL
00011B46 E86AFFFFFF          <1>       call   pciRegRead8
                             <1>       ; ?
00011B4B B040                <1>       mov    al, VIA_ACLINK_STAT
00011B4D E863FFFFFF          <1>       call   pciRegRead8
00011B52 F6C201              <1>       test   dl, VIA_ACLINK_C00_READY
00011B55 7508                <1>         jnz     short _codec_ready_1
00011B57 E80E000000          <1>       call   reset_codec
00011B5C 7306                <1>       jnc    short _codec_ready_2 ; eax = 1
00011B5E C3                  <1>       retn
                             <1> _codec_ready_1:
00011B5F B801000000          <1>       mov    eax, 1
                             <1> _codec_ready_2:
00011B64 E87A000000          <1>       call   codec_io_w16
                             <1> detect_codec:
00011B69 C3                  <1>       retn
                             <1>
                             <1> reset_codec:
                             <1>       ; 16/04/2017
                             <1>       ; 23/03/2017
                             <1>       ; ('codec.asm')
                             <1>       ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
00011B6A A1[C46B0100]        <1>       mov    eax, [audio_dev_id]
00011B6F B041                <1>       mov    al, VIA_ACLINK_CTRL
00011B71 B2E0                <1>             mov    dl, VIA_ACLINK_CTRL_ENABLE + VIA_ACLINK_CTRL_RESET + VIA_ACLINK_CTRL_SYNC
00011B73 E8A8FFFFFF          <1>       call   pciRegWrite8
                             <1>
00011B78 E83D000000          <1>       call   delay_100ms  ; wait 100 ms
```

```
                             <1> _rc_cold:
00011B7D E808000000          <1>         call    cold_reset
00011B82 7301                <1>         jnc     short _reset_codec_ok
                             <1> 
                             <1>       ; 16/04/2017
                             <1>         ;xor     eax, eax         ; timeout error
                             <1>             ;stc
00011B84 C3                  <1>       retn
                             <1> 
                             <1> _reset_codec_ok:
00011B85 31C0                <1>         xor     eax, eax
                             <1>         ;mov al, VIA_ACLINK_C00_READY ; 1
00011B87 FEC0                <1>         inc al
00011B89 C3                  <1>       retn
                             <1> 
                             <1> cold_reset:
                             <1>       ; 16/04/2017
                             <1>       ; 23/03/2017
                             <1>       ; ('codec.asm')
                             <1>       ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
                             <1>       ;mov  eax, [audio_dev_id]
                             <1>       ;mov  al, VIA_ACLINK_CTRL
00011B8A 30D2                <1>       xor   dl, dl ; 0
00011B8C E88FFFFFFF          <1>       call  pciRegWrite8
                             <1> 
00011B91 E824000000          <1>       call  delay_100ms  ; wait 100 ms
                             <1> 
                             <1>       ;; ACLink on, deassert ACLink reset, VSR, SGD data out
                             <1>         ;; note - FM data out has trouble with non VRA codecs !!
                             <1> 
                             <1>       ;mov  eax, [audio_dev_id]
                             <1>       ;mov  al, VIA_ACLINK_CTRL
00011B96 B2CC                <1>       mov   dl, VIA_ACLINK_CTRL_INIT
00011B98 E883FFFFFF          <1>       call  pciRegWrite8
                             <1> 
00011B9D B910000000          <1>       mov   ecx, 16      ; total 2s
                             <1> 
                             <1> _crst_wait:
                             <1>       ;mov  eax, [audio_dev_id]
00011BA2 B040                <1>       mov   al, VIA_ACLINK_STAT
00011BA4 E80CFFFFFF          <1>       call  pciRegRead8
                             <1> 
00011BA9 F6C201              <1>         test    dl, VIA_ACLINK_C00_READY
00011BAC 750B                <1>         jnz     short _crst_ok
                             <1> 
00011BAE 51                  <1>       push  ecx
00011BAF E806000000          <1>       call  delay_100ms
00011BB4 59                  <1>       pop   ecx
                             <1> 
00011BB5 49                  <1>         dec     ecx
00011BB6 75EA                <1>         jnz     short _crst_wait
                             <1> 
                             <1> _crst_fail:
00011BB8 F9                  <1>         stc
                             <1> _crst_ok:
00011BB9 C3                  <1>       retn
                             <1> 
                             <1> delay_100ms:
                             <1>       ; 29/05/2017
                             <1>       ; 24/03/2017 ('codec.asm')
                             <1>       ; wait 100 ms
00011BBA B990010000          <1>       mov   ecx, 400  ; 400*0.25ms
                             <1> _delay_x_ms:
00011BBF E803000000          <1>       call  delay1_4ms
00011BC4 E2F9                <1>         loop _delay_x_ms
00011BC6 C3                  <1>       retn
                             <1> 
                             <1> ;       delay1_4ms - Delay for 1/4 millisecond.
                             <1> ;         1mS = 1000us
                             <1> ;       Entry:
                             <1> ;         None
                             <1> ;       Exit:
                             <1> ;       None
                             <1> ;
                             <1> ;       Modified:
                             <1> ;         None
                             <1> ;
                             <1> 
                             <1>       ; 29/05/2017
                             <1>       ; 23/04/2017
                             <1>       ; 05/03/2017 (TRDOS 386)
                             <1>       ; ('UTILS.ASM')
                             <1> delay1_4ms:
00011BC7 50                  <1>         push    eax
00011BC8 51                  <1>         push    ecx
00011BC9 B110                <1>         mov cl, 16       ; close enough.
                             <1> 
00011BCB E461                <1>       in    al, PORTB ; 61h
                             <1> 
00011BCD 2410                <1>       and   al, REFRESH_STATUS ; 10h
00011BCF 88C5                <1>       mov   ch, al       ; Start toggle state
                             <1> _d4ms1:
00011BD1 E461                <1>       in    al, PORTB    ; Read system control port
                             <1> 
00011BD3 2410                <1>       and   al, REFRESH_STATUS ; Refresh toggles 15.085 microseconds
00011BD5 38C5                <1>       cmp   ch, al
00011BD7 74F8                <1>       je    short _d4ms1 ; Wait for state change
                             <1> 
00011BD9 88C5                <1>       mov   ch, al       ; Update with new state
00011BDB FEC9                <1>       dec   cl
00011BDD 75F2                <1>       jnz   short _d4ms1
                             <1> 
00011BDF F8                  <1>       clc   ; 29/05/2017
```

```
                             <1> 
00011BE0 59                  <1>         pop     ecx
00011BE1 58                  <1>         pop     eax
00011BE2 C3                  <1>         retn
                             <1> 
                             <1> ; 10/04/2017 (TRDOS 386)
                             <1> ; 12/11/2016
                             <1> 
                             <1> codec_io_w16: ;w32
                             <1>       ; ('codec.asm')
00011BE3 668B15[C26B0100]    <1>         mov dx, [audio_io_base]
00011BEA 6681C28000          <1>         add     dx, VIA_REG_AC97
00011BEF EF                  <1>       out   dx, eax
00011BF0 C3                  <1>         retn
                             <1> 
                             <1> codec_io_r16: ;r32
                             <1>       ; ('codec.asm')
00011BF1 668B15[C26B0100]    <1>         mov     dx, [audio_io_base]
00011BF8 6681C28000          <1>         add     dx, VIA_REG_AC97
00011BFD ED                  <1>         in  eax, dx
00011BFE C3                  <1>         retn
                             <1> 
                             <1> ctrl_io_w8:
                             <1>       ; ('codec.asm')
00011BFF 660315[C26B0100]    <1>         add     dx, [audio_io_base]
00011C06 EE                  <1>         out dx, al
00011C07 C3                  <1>         retn
                             <1> 
                             <1> ctrl_io_r8:
                             <1>       ; ('codec.asm')
00011C08 660315[C26B0100]    <1>         add     dx, [audio_io_base]
00011C0F EC                  <1>         in  al, dx
00011C10 C3                  <1>         retn
                             <1> 
                             <1> ctrl_io_w32:
                             <1>       ; ('codec.asm')
00011C11 660315[C26B0100]    <1>         add     dx, [audio_io_base]
00011C18 EF                  <1>         out dx, eax
00011C19 C3                  <1>         retn
                             <1> 
                             <1> ctrl_io_r32:
                             <1>       ; ('codec.asm')
00011C1A 660315[C26B0100]    <1>         add dx, [audio_io_base]
00011C21 ED                  <1>       in    eax, dx
00011C22 C3                  <1>         retn
                             <1> 
                             <1> codec_read:
                             <1>       ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
                             <1>         ; Use only primary codec.
                             <1>         ; eax = register
00011C23 C1E010              <1>         shl     eax, VIA_REG_AC97_CMD_SHIFT
00011C26 0D00008002          <1>         or      eax, VIA_REG_AC97_PRIMARY_VALID + VIA_REG_AC97_READ
                             <1> 
00011C2B E8B3FFFFFF          <1>       call    codec_io_w16
                             <1> 
                             <1>             ; codec_valid
00011C30 E831000000          <1>       call  codec_check_ready
00011C35 7301                <1>         jnc short _cr_ok
                             <1> 
00011C37 C3                  <1>       retn
                             <1> 
                             <1> _cr_ok:
                             <1>       ; wait 25 ms
00011C38 B950000000          <1>       mov   ecx, 80 ; (100*0.25 ms)
                             <1> _cr_wloop:
00011C3D E885FFFFFF          <1>       call  delay1_4ms
00011C42 E2F9                <1>       loop  _cr_wloop
                             <1> 
00011C44 E8A8FFFFFF          <1>         call    codec_io_r16
00011C49 25FFFF0000          <1>         and     eax, 0FFFFh
00011C4E C3                  <1>         retn
                             <1> 
                             <1> codec_write:
                             <1>       ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
                             <1>         ; Use only primary codec.
                             <1> 
                             <1>       ; eax = data (volume)
                             <1>       ; edx = register (mixer register)
                             <1> 
00011C4F C1E210              <1>       shl     edx, VIA_REG_AC97_CMD_SHIFT
                             <1> 
00011C52 C1E000              <1>         shl     eax, VIA_REG_AC97_DATA_SHIFT ; shl eax, 0
00011C55 09C2                <1>         or      edx, eax
                             <1> 
00011C57 B800000000          <1>         mov     eax, VIA_REG_AC97_CODEC_ID_PRIMARY
00011C5C C1E01E              <1>         shl     eax, VIA_REG_AC97_CODEC_ID_SHIFT
00011C5F 09D0                <1>         or      eax, edx
                             <1> 
00011C61 E87DFFFFFF          <1>         call    codec_io_w16
                             <1>         ;mov    [codec.regs+esi], ax
                             <1> 
                             <1>         ;call      codec_check_ready
                             <1>             ;retn
                             <1>       ;jmp  short _codec_check_ready
                             <1> 
                             <1> codec_check_ready:
                             <1>       ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
                             <1> 
                             <1> _codec_check_ready:
00011C66 B914000000          <1>       mov   ecx, 20       ; total 2s
                             <1> _ccr_wait:
00011C6B 51                  <1>       push  ecx
                             <1> 
```

```
00011C6C E880FFFFFF           <1>         call    codec_io_r16
00011C71 A900000001           <1>         test    eax, VIA_REG_AC97_BUSY
00011C76 740B                 <1>         jz      short _ccr_ok
                              <1>
00011C78 E83DFFFFFF           <1>       call   delay_100ms
                              <1>
00011C7D 59                   <1>       pop    ecx
                              <1>
00011C7E 49                   <1>       dec     ecx
00011C7F 75EA                 <1>         jnz     short _ccr_wait
                              <1>
00011C81 F9                   <1>         stc
00011C82 C3                   <1>         retn
                              <1>
                              <1> _ccr_ok:
00011C83 59                   <1>       pop    ecx
00011C84 25FFFF0000           <1>       and     eax, 0FFFFh
00011C89 C3                   <1>         retn
                              <1>
                              <1> codec_config:
                              <1>       ; 10/06/2017
                              <1>       ; 29/05/2017
                              <1>       ; 24/04/2017
                              <1>       ; 21/04/2017
                              <1>       ; 16/04/2017 (TRDOS 386 Kernel)
                              <1>       ; 15/11/2016 ('codec.asm', 'player.com')
                              <1>       ; 14/11/2016
                              <1>       ; 12/11/2016 - Erdogan Tan
                              <1>       ;          (Ref: KolibriOS, 'setup_codec', codec.inc)
                              <1>
00011C8A B802020000           <1>       mov     eax, 0202h
00011C8F 66A3[F26B0100]       <1>       mov    [audio_master_volume], ax
00011C95 66B81F1F             <1>       mov    ax, 1F1Fh ; 31,31
00011C99 BA02000000           <1>       mov    edx, CODEC_MASTER_VOL_REG ; 02h ; Line Out
00011C9E E8ACFFFFFF           <1>       call   codec_write
                              <1>       ;jc    short cconfig_error
                              <1>
                              <1>       ;mov    eax, 0202h
00011CA3 66B80202             <1>       mov     ax, 0202h
00011CA7 BA18000000           <1>       mov    edx, CODEC_PCM_OUT_REG ; 18h ; Wave Output (Stereo)
00011CAC E89EFFFFFF           <1>       call   codec_write
                              <1>       ;jc    short cconfig_error
                              <1>
                              <1>       ;mov    eax, 0202h
00011CB1 66B80202             <1>       mov    ax, 0202h
00011CB5 BA04000000           <1>       mov    edx, CODEC_AUX_VOL ; 04h ; CODEC_HP_VOL_REG ; HeadPhone
00011CBA E890FFFFFF           <1>       call   codec_write
                              <1>       ;jc    short cconfig_error
                              <1>
                              <1>       ;mov    eax, 08h
                              <1>         ;mov    ax,  08h
00011CBF 66B80880             <1>       mov    ax, 8008h ; Mute
00011CC3 BA0C000000           <1>       mov    edx, 0Ch  ; AC97_PHONE_VOL ; TAD Input (Mono)
00011CC8 E882FFFFFF           <1>       call   codec_write
                              <1>       ;jc    short cconfig_error
                              <1>
                              <1>       ;mov    eax, 0808h
00011CCD 66B80808             <1>         mov     ax,  0808h
00011CD1 BA10000000           <1>         mov edx, CODEC_LINE_IN_VOL_REG ; 10h ; Line Input (Stereo)
00011CD6 E874FFFFFF           <1>       call   codec_write
                              <1>       ;jc    short cconfig_error
                              <1>
                              <1>       ;mov    eax, 0808h
00011CDB 66B80808             <1>       mov     ax,  0808h
00011CDF BA12000000           <1>         mov edx, CODEC_CD_VOL_REG ; 12h ; CR Input (Stereo)
00011CE4 E866FFFFFF           <1>       call   codec_write
                              <1>       ;jc    short cconfig_error
                              <1>
                              <1>       ;mov    eax, 0808h
00011CE9 66B80808             <1>       mov     ax,  0808h
00011CED BA16000000           <1>         mov edx, CODEC_AUX_VOL_REG ; 16h ; Aux Input (Stereo)
                              <1>       ;call  codec_write
                              <1>       ;;jc   short cconfig_error
00011CF2 E958FFFFFF           <1>       jmp    codec_write ; 10/06/2017
                              <1>
                              <1> ;     ; Extended Audio Status (2Ah)
                              <1> ;     mov    eax, CODEC_EXT_AUDIO_CTRL_REG ; 2Ah
                              <1> ;     call   codec_read
                              <1> ;     and     eax, 0FFFFh - 2          ; clear DRA (BIT1)
                              <1> ;     ;or     eax, 1                   ; set VRA (BIT0)
                              <1> ;     or     eax, 5        ; VRA (BIT0) & S/PDIF (BIT2) ; 14/11/2016
                              <1> ;     mov    edx, CODEC_EXT_AUDIO_CTRL_REG
                              <1> ;     call   codec_write
                              <1> ;     ;jc    short cconfig_error
                              <1> ;
                              <1> ;set_sample_rate:
                              <1> ;     ;movzx eax, word [audio_freq]
                              <1> ;     mov    ax, [audio_freq]
                              <1> ;     mov    edx, CODEC_PCM_FRONT_DACRATE_REG ; 2Ch ; PCM Front DAC Rate
                              <1> ;     ;call  codec_write
                              <1> ;     ;retn
                              <1> ;     jmp    codec_write
                              <1>
                              <1> ;cconfig_error:
                              <1> ;     retn
                              <1>
                              <1> vt8233_int_handler:
                              <1>       ; Interrupt Handler for VIA VT8237R Audio Controller
                              <1>       ; Note: called by 'dev_IRQ_service'
                              <1>       ; 14/10/2017
                              <1>       ; 09/10/2017, 10/10/2017, 12/10/2017
                              <1>       ; 13/06/2017
                              <1>       ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
```

```
                             <1>       ; 24/03/2017 - 'PLAYER.COM' ('player.asm')
                             <1>
                             <1>       ;push  eax ; * must be saved !
                             <1>       ;push  edx
                             <1>       ;push  ecx
                             <1>       ;push  ebx ; * must be saved !
                             <1>       ;push  esi
                             <1>       ;push  edi
                             <1>
                             <1>       ;cmp   byte [audio_busy], 1
                             <1>       ;jnb   short _ih0 ; 09/10/2017
                             <1>
                             <1>       ;mov   byte [audio_flag_eol], 0
                             <1>
00011CF7 66BA0000            <1>         mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STATUS
00011CFB E808FFFFFF          <1>         call    ctrl_io_r8
                             <1>
00011D00 A880                <1>       test    al, VIA_REG_STAT_ACTIVE
00011D02 7417                <1>         jz      short _ih0 ; 09/10/2017
                             <1>
00011D04 2407                <1>         and     al, VIA_REG_STAT_EOL + VIA_REG_STAT_FLAG + VIA_REG_STAT_STOPPED
00011D06 A2[F16B0100]        <1>       mov    [audio_flag_eol], al
00011D0B 740E                <1>         jz   short _ih0 ; 09/10/2017
                             <1>
                             <1>       ; 09/10/2017
                             <1>       ;mov   byte [audio_busy], 1
                             <1>
00011D0D 803D[F06B0100]01    <1>       cmp    byte [audio_play_cmd], 1
00011D14 7315                <1>       jnb    short _ih1 ; 10/10/2017
                             <1>
00011D16 E860000000          <1>       call   channel_reset
                             <1> _ih0:
                             <1>       ; 09/10/2017
00011D1B A0[F16B0100]        <1>         mov     al, [audio_flag_eol]   ;; ack ;;
00011D20 66BA0000            <1>         mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STATUS
00011D24 E8D6FEFFFF          <1>         call    ctrl_io_w8
00011D29 EB4F                <1>       jmp    short _ih4
                             <1> _ih1:
                             <1> vt8233_tuneLoop:
00011D2B A0[F16B0100]        <1>         mov     al, [audio_flag_eol]   ;; ack ;;
00011D30 66BA0000            <1>         mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STATUS
00011D34 E8C6FEFFFF          <1>         call    ctrl_io_w8
                             <1>
                             <1>       ; 12/10/2017
00011D39 C605[E46B0100]00    <1>       mov    byte [audio_flag], 0 ; Reset
                             <1>
                             <1>       ; 10/10/2017
                             <1>       ; 09/10/2017
                             <1>       ;test  byte [audio_flag_eol], VIA_REG_STAT_FLAG
                             <1>       ;jz    short _ih2 ; EOL
                             <1>
                             <1>       ; 14/10/2017
00011D40 F605[F16B0100]02    <1>       test   byte [audio_flag_eol], VIA_REG_STAT_EOL
00011D47 7506                <1>       jnz    short _ih2 ; EOL
                             <1>                       ; (Half Buffer 2 has been completed
                             <1>                       ; and Half Buffer 1 will be played.)
                             <1>       ; FLAG
                             <1>       ; (Half Buffer 1 has been completed
                             <1>       ;  and Half Buffer 2 will be played.)
                             <1>
                             <1>       ; 14/10/2017
                             <1>       ;; (Continue to play.)
                             <1>       ;mov   al, VIA_REG_CTRL_INT
                             <1>             ;or    al, VIA_REG_CTRL_START
                             <1>             ;mov   dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
                             <1>         ;call      ctrl_io_w8
                             <1>       ; 12/10/2017
                             <1>       ;mov   byte [audio_flag], 1
00011D49 FE05[E46B0100]      <1>       inc    byte [audio_flag] ; = 1
                             <1> _ih2:
                             <1>       ; 10/10/2017
00011D4F 8B3D[DC6B0100]      <1>       mov    edi, [audio_dma_buff]
00011D55 8B0D[E06B0100]      <1>       mov    ecx, [audio_dmabuff_size]
00011D5B D1E9                <1>       shr    ecx, 1 ; dma buff size / 2 = half buffer size
                             <1>
                             <1>       ; 12/10/2017
00011D5D 803D[E46B0100]00    <1>       cmp    byte [audio_flag], 0
00011D64 7702                <1>       ja     short _ih3 ; Playing Half Buffer 2 (Current: FLAG)
                             <1>       ; Playing Half Buffer 1 (Current: EOL)
00011D66 01CF                <1>       add    edi, ecx
                             <1> _ih3:
                             <1>       ; Update half buffer 2 while playing half buffer 1 (FLAG)
                             <1>       ; Update half buffer 1 while playing half buffer 2 (EOL)
                             <1>
00011D68 8B35[D46B0100]      <1>       mov    esi, [audio_p_buffer] ; phy addr of audio buff
00011D6E C1E902              <1>       shr    ecx, 2 ; half buff size / 4
00011D71 F3A5                <1>       rep    movsd
                             <1>       ; switch flag value ;
00011D73 8035[E46B0100]01    <1>       xor    byte [audio_flag], 1 ; 10/10/2017
                             <1>       ; 12/10/2017
                             <1>       ; [audio_flag] = 0 : Playing dma half buffer 2 (just after FLAG)
                             <1>                       ; Next buffer (to update) is dma half buff 1
                             <1>       ;             = 1 : Playing dma half buffer 1 (just after EOL)
                             <1>                       ; Next buffer (to update) is dma half buff 2
                             <1> _ih4:
                             <1>       ; 28/05/2017
                             <1>       ;mov   byte [audio_busy], 0 ; 09/10/2017
                             <1>       ;
                             <1>       ;pop   edi
                             <1>       ;pop   esi
                             <1>       ;pop   ebx ; * must be restored !
                             <1>       ;pop   ecx
                             <1>       ;pop   edx
                             <1>       ;pop   eax ; * must be restored !
                             <1> 
00011D7A C3                  <1>       retn
                             <1> 
                             <1> channel_reset:
                             <1>       ; 24/06/2017
                             <1>       ; 29/05/2017
                             <1>       ; 23/03/2017
                             <1>       ; 14/11/2016 - Erdogan Tan
                             <1>       ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
00011D7B BA01000000          <1>         mov  edx, VIA_REG_OFFSET_CONTROL
                             <1>         ;mov eax, VIA_REG_CTRL_PAUSE + VIA_REG_CTRL_TERMINATE + VIA_REG_CTRL_RESET
00011D80 B848000000          <1>         mov  eax, VIA_REG_CTRL_PAUSE + VIA_REG_CTRL_TERMINATE ; 24/06/2017
00011D85 E875FEFFFF          <1>       call    ctrl_io_w8
                             <1> 
                             <1>         ;mov edx, VIA_REG_OFFSET_CONTROL
                             <1>         ;call   ctrl_io_r8
                             <1> 
                             <1>       ; wait for 50 ms
00011D8A B9A0000000          <1>       mov    ecx, 160 ; (200*0.25 ms) ; 29/05/2017
                             <1> _ch_rst_wait:
00011D8F E833FEFFFF          <1>       call   delay1_4ms
00011D94 49                  <1>       dec    ecx
00011D95 75F8                <1>       jnz    short _ch_rst_wait
                             <1> 
                             <1>         ; disable interrupts
00011D97 BA01000000          <1>         mov  edx, VIA_REG_OFFSET_CONTROL
00011D9C 31C0                <1>         xor     eax, eax
00011D9E E85CFEFFFF          <1>         call    ctrl_io_w8
                             <1> 
                             <1>         ; clear interrupts
00011DA3 BA00000000          <1>         mov  edx, VIA_REG_OFFSET_STATUS
00011DA8 B803000000          <1>       mov    eax, 3
00011DAD E84DFEFFFF          <1>         call ctrl_io_w8
                             <1> 
                             <1>       ;mov   edx, VIA_REG_OFFSET_CURR_PTR
                             <1>       ;xor   eax, eax
                             <1>       ;call  ctrl_io_w32
                             <1> 
00011DB2 C3                  <1>         retn
                             <1> 
                             <1> vt8233_stop: ; 22/04/2017
00011DB3 C605[F06B0100]00    <1>       mov    byte [audio_play_cmd], 0 ; stop !
                             <1> _tlp2:
                             <1>       ; 24/06/2017
                             <1>         ; finished with song, stop everything
                             <1>       ;mov   al, VIA_REG_CTRL_INT
                             <1>         ;or  al, VIA_REG_CTRL_TERMINATE
                             <1>       ;mov   dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
                             <1>         ;call      ctrl_io_w8
                             <1> 
                             <1>         ;call      channel_reset
                             <1>       ;retn
00011DBA EBBF                <1>       jmp    short channel_reset
                             <1> 
                             <1> set_vt8233_bdl: ; Set VT8237R Buffer Descriptor List
                             <1>       ; 28/05/2017
                             <1>       ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
                             <1>       ; 24/03/2017 - 'PLAYER.COM' ('via_wav.asm' - 29/11/2016)
                             <1> 
                             <1>       ; eax = dma buffer address = [audio_DMA_buff]
                             <1>       ; ecx = dma buffer buffer size = [audio_dmabuff_size]
                             <1> 
00011DBC D1E9                <1>       shr    ecx, 1 ; dma half buffer size
00011DBE 89CE                <1>       mov    esi, ecx
                             <1> 
00011DC0 BF[F86B0100]        <1>         mov     edi, audio_bdl_buff    ; get BDL address
00011DC5 B910000000          <1>         mov     ecx, 32 / 2            ; make 32 entries in BDL
                             <1> 
00011DCA EB05                <1>       jmp    short s_vt8233_bdl1
                             <1> 
                             <1> s_vt8233_bdl0:
                             <1>       ; set buffer descriptor 0 to start of data file in memory
                             <1> 
00011DCC A1[DC6B0100]        <1>       mov    eax, [audio_dma_buff]     ; Physical address of DMA buffer
                             <1> 
                             <1> s_vt8233_bdl1:
00011DD1 AB                  <1>       stosd                     ; store dmabuffer1 address
                             <1> 
00011DD2 89C2                <1>       mov    edx, eax
                             <1> 
                             <1> ; VIA VT8235.PDF: (Page 110) (Erdogan Tan, 29/11/2016)
                             <1>       ;
                             <1>       ;      Audio SGD Table Format
                             <1>       ;      -------------------------------
                             <1>       ;      63   62    61-56    55-32   31-0
                             <1>       ;      --   --   --------  -----   ----
                             <1>       ;      EOL FLAG -reserved- Base    Base
                             <1>       ;                        Count  Address
                             <1>       ;                         [23:0] [31:0]
                             <1>       ;      EOL: End Of Link.
                             <1>       ;           1 indicates this block is the last of the link.
                             <1>       ;           If the channel “Interrupt on EOL” bit is set, then
                             <1>       ;           an interrupt is generated at the end of the transfer.
                             <1>       ;
                             <1>       ;      FLAG: Block Flag. If set, transfer pauses at the end of this
                             <1>       ;            block. If the channel “Interrupt on FLAG” bit is set,
                             <1>       ;            then an interrupt is generated at the end of this block.
                             <1> 
00011DD4 89F0                <1>       mov    eax, esi ; DMA half buffer size
00011DD6 01C2                <1>       add    edx, eax
00011DD8 0D00000040          <1>       or     eax, FLAG
                             <1>       ;or    eax, EOL
```

```
00011DDD AB                  <1>       stosd
                             <1>
                             <1> ; 2nd buffer:
                             <1>
00011DDE 89D0                <1>         mov  eax, edx ; Physical address of the 2nd half of DMA buffer
00011DE0 AB                  <1>       stosd         ; store dmabuffer2 address
                             <1>
                             <1> ; set length to [audio_dmabuff_size]/2
                             <1> ; Set control (bits 31:16) to BUP, bits 15:0=number of samples
                             <1> ;
00011DE1 89F0                <1>       mov    eax, esi ; DMA half buffer size
00011DE3 0D00000080          <1>       or     eax, EOL
                             <1>       ;or    eax, FLAG
00011DE8 AB                  <1>       stosd
                             <1>
00011DE9 E2E1                <1>       loop    s_vt8233_bdl0
                             <1>
00011DEB C3                  <1>       retn
                             <1>
                             <1> vt8233_start_play:
                             <1>       ; start to play audio data via VT8233 audio controller
                             <1>       ; 13/06/2017
                             <1>       ; 10/06/2017
                             <1>       ; 24/04/2017
                             <1>       ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
                             <1>       ; 24/03/2017 - 'PLAYER.COM' ('via_wav.asm' - 29/11/2016)
                             <1>       ; write buffer descriptor list address
                             <1>       ;
                             <1>
                             <1>       ; Extended Audio Status (2Ah)
00011DEC B82A000000          <1>       mov    eax, CODEC_EXT_AUDIO_CTRL_REG ; 2Ah
00011DF1 E82DFEFFFF          <1>       call   codec_read
00011DF6 25FDFF0000          <1>       and     eax, 0FFFFh - 2          ; clear DRA (BIT1)
                             <1>       ;or     eax, 1                   ; set VRA (BIT0)
00011DFB 83C805              <1>       or     eax, 5       ; VRA (BIT0) & S/PDIF (BIT2) ; 14/11/2016
00011DFE BA2A000000          <1>       mov    edx, CODEC_EXT_AUDIO_CTRL_REG
00011E03 E847FEFFFF          <1>       call   codec_write
                             <1>       ;jc    short cconfig_error
                             <1>
                             <1> set_sample_rate:
                             <1>       ;movzx eax, word [audio_freq]
00011E08 66A1[EE6B0100]      <1>       mov    ax, [audio_freq]
00011E0E BA2C000000          <1>       mov    edx, CODEC_PCM_FRONT_DACRATE_REG ; 2Ch ; PCM Front DAC Rate
00011E13 E837FEFFFF          <1>       call   codec_write
                             <1>
00011E18 B8[F86B0100]        <1>         mov  eax, audio_bdl_buff
                             <1>
                             <1>       ; 12/11/2016 - Erdogan Tan
                             <1>       ; (Ref: KolibriOS, vt823x.asm, 'create_primary_buff')
00011E1D BA04000000          <1>       mov    edx, VIADEV_PLAYBACK + VIA_REG_OFFSET_TABLE_PTR
00011E22 E8EAFDFFFF          <1>         call ctrl_io_w32
                             <1>
                             <1>       ;call  codec_check_ready
                             <1>
00011E27 66BA0200            <1>       mov    dx, VIADEV_PLAYBACK + VIA_REG_OFS_PLAYBACK_VOLUME_L
                             <1>         ;mov eax, 2 ; 31
00011E2B B01F                <1>       mov    al, 31
00011E2D 2A05[F26B0100]      <1>         sub  al, [audio_master_volume_l]
00011E33 E8C7FDFFFF          <1>       call   ctrl_io_w8
                             <1>
                             <1>       ;call  codec_check_ready
                             <1>
00011E38 66BA0300            <1>         mov     dx, VIADEV_PLAYBACK + VIA_REG_OFS_PLAYBACK_VOLUME_R
                             <1>         ;mov ax, 2  ; 31
00011E3C B01F                <1>       mov    al, 31
00011E3E 2A05[F36B0100]      <1>         sub  al, [audio_master_volume_r]
00011E44 E8B6FDFFFF          <1>       call    ctrl_io_w8
                             <1>
                             <1>       ;call  codec_check_ready
                             <1> ;
                             <1> ;
                             <1> ; All set.  Let's play some music.
                             <1> ;
                             <1> ;
                             <1>             ;mov    dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STOP_IDX
                             <1>         ;mov    ax, VIA8233_REG_TYPE_16BIT or VIA8233_REG_TYPE_STEREO or 0xfffff or 0xff000000
                             <1>         ;call   ctrl_io_w32
                             <1>
                             <1>       ;call  codec_check_ready
                             <1>
                             <1>       ; 08/12/2016
                             <1>       ; 07/10/2016
                             <1>         ;mov    al, 1
00011E49 B01F                <1>         mov  al, 31
00011E4B E815000000          <1>       call    set_VT8233_LastValidIndex
                             <1>
00011E50 C605[F06B0100]01    <1>       mov    byte [audio_play_cmd], 1 ; play command (do not stop) !
                             <1>
                             <1> vt8233_play: ; continue to play
                             <1>       ; 22/04/2017
00011E57 B023                <1>         mov  al, VIA_REG_CTRL_INT
00011E59 0C80                <1>             or     al, VIA_REG_CTRL_START
                             <1>         ;mov al, VIA_REG_CTRL_AUTOSTART + VIA_REG_CTRL_START
                             <1>       ;mov   al, VIA_REG_CTRL_AUTOSTART + VIA_REG_CTRL_START + VIA_REG_CTRL_INT_FLAG
00011E5B 66BA0100            <1>       mov    dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
00011E5F E89BFDFFFF          <1>         call    ctrl_io_w8
                             <1>       ;call  codec_check_ready
                             <1>       ;retn
                             <1>       ;jmp   codec_check_ready
00011E64 C3                  <1>       retn
                             <1>
                             <1> ;input AL = index # to stop on
                             <1> set_VT8233_LastValidIndex:
                             <1>       ; 10/06/2017
                             <1>       ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
                             <1>       ; 24/03/2017 - 'PLAYER.COM' ('via_wav.asm' - 29/11/2016)
                             <1>       ; 19/11/2016
                             <1>       ; 14/11/2016 - Erdogan Tan (Ref: VIA VT8235.PDF, Page 110)
                             <1>       ; 12/11/2016 - Erdogan Tan
                             <1>       ; (Ref: KolibriOS, vt823x.asm, 'create_primary_buff')
                             <1>       ;push  edx
00011E65 6650                    <1>       push  ax
                             <1>       ;push  ecx
00011E67 0FB705[EE6B0100]        <1>       movzx eax, word [audio_freq] ; Hertz
00011E6E BA00001000              <1>       mov   edx, 100000h ; 2^20 = 1048576
00011E73 F7E2                    <1>       mul   edx
00011E75 B980BB0000              <1>       mov   ecx, 48000
00011E7A F7F1                    <1>       div   ecx
                             <1>       ;and  eax, 0FFFFFh
                             <1>       ;pop  ecx
00011E7C 665A                    <1>       pop   dx
00011E7E C1E218                  <1>       shl   edx, 24  ; STOP Index Setting: Bit 24 to 31
00011E81 09D0                    <1>       or    eax, edx
                             <1>       ; 19/11/2016
00011E83 803D[EC6B0100]10        <1>       cmp   byte [audio_bps], 16
00011E8A 7505                    <1>       jne   short sLVI_1
00011E8C 0D00002000              <1>       or    eax, VIA8233_REG_TYPE_16BIT
                             <1> sLVI_1:
00011E91 803D[ED6B0100]02        <1>       cmp   byte [audio_stmo], 2
00011E98 7505                    <1>       jne   short sLVI_2
00011E9A 0D00001000              <1>       or    eax, VIA8233_REG_TYPE_STEREO
                             <1> sLVI_2:
00011E9F BA08000000              <1>       mov     edx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STOP_IDX
00011EA4 E868FDFFFF              <1>         call    ctrl_io_w32
                             <1>       ;call  codec_check_ready
                             <1>       ;pop   edx
00011EA9 C3                      <1>       retn
                             <1>
                             <1> vt8233_pause: ; pause
                             <1>       ; 10/06/2017
                             <1>       ; 22/04/2017
00011EAA B023                    <1>       mov     al, VIA_REG_CTRL_INT
00011EAC 0C08                    <1>         or      al, VIA_REG_CTRL_PAUSE
00011EAE 66BA0100                <1>       mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
00011EB2 E848FDFFFF              <1>         call    ctrl_io_w8
                             <1>       ;call  codec_check_ready
                             <1>       ;retn
                             <1>       ;jmp   codec_check_ready
00011EB7 C3                      <1>       retn
                             <1>
                             <1> vt8233_reset:
                             <1>       ; 22/04/2017
                             <1>       ; reset VT8237R (vt8233) Audio Controller
                             <1>       ;cmp   byte [audio_play_cmd], 1
                             <1>       ;jna   short vt8233_rst_0
00011EB8 C605[F06B0100]00        <1>       mov   byte [audio_play_cmd], 0 ; stop !
                             <1> vt8233_rst_0:
00011EBF E8A6FCFFFF              <1>       call  reset_codec
00011EC4 720A                    <1>       jc    short vt8233_rst_1 ; codec error !
                             <1>       ; eax = 1
00011EC6 E818FDFFFF              <1>       call  codec_io_w16 ; w32
00011ECB E8ABFEFFFF              <1>       call  channel_reset
                             <1> vt8233_rst_1:
00011ED0 C3                      <1>       retn
                             <1>
                             <1> vt8233_volume:
                             <1>       ; set VT8237R (vt8233) sound volume level
                             <1>       ; 24/04/2017
                             <1>       ; 22/04/2017
                             <1>       ; bl = component (0 = master/playback/lineout volume)
                             <1>       ; cl = left channel volume level (0 to 31)
                             <1>       ; ch = right channel volume level (0 to 31)
                             <1>
00011ED1 08DB                    <1>       or    bl, bl
00011ED3 7520                    <1>       jnz   short vt8233_vol_1 ; temporary !
00011ED5 66B81F1F                <1>       mov   ax, 1F1Fh ; 31,31
00011ED9 38C1                    <1>       cmp   cl, al
00011EDB 7718                    <1>       ja    short vt8233_vol_1 ; temporary !
00011EDD 38E5                    <1>       cmp   ch, ah
00011EDF 7714                    <1>       ja    short vt8233_vol_1 ; temporary !
00011EE1 66890D[F26B0100]        <1>       mov   [audio_master_volume], cx
00011EE8 6629C8                  <1>       sub   ax, cx
00011EEB BA02000000              <1>       mov   edx, CODEC_MASTER_VOL_REG ; 02h ; Line Out
00011EF0 E85AFDFFFF              <1>       call  codec_write
                             <1> vt8233_vol_1:
00011EF5 C3                      <1>       retn
                             <1>
                             <1> ; CODE for SOUND BLASTER 16
                             <1>
                             <1> DetectSB:
                             <1>       ; 24/04/2017
                             <1>       ;pushad
                             <1> ScanPort:
00011EF6 66BB1002                <1>       mov     bx, 210h    ; start scanning ports
                             <1>                           ; 210h, 220h, .. 260h
                             <1> ResetDSP:
00011EFA 6689DA                  <1>       mov     dx, bx             ; try to reset the DSP.
00011EFD 6683C206                <1>       add     dx, 06h
00011F01 B001                    <1>       mov    al, 1
00011F03 EE                      <1>       out    dx, al
                             <1>
00011F04 EC                      <1>       in     al, dx
00011F05 EC                      <1>       in     al, dx
00011F06 EC                      <1>       in     al, dx
00011F07 EC                      <1>       in     al, dx
                             <1>
```

```
00011F08 30C0                <1>        xor     al, al
00011F0A EE                  <1>        out    dx, al
                             <1> 
00011F0B 6683C208            <1>        add     dx, 08h
00011F0F 66B96400            <1>        mov    cx, 100
                             <1> WaitID:
00011F13 EC                  <1>        in     al, dx
00011F14 08C0                <1>        or      al, al
00011F16 7804                <1>        js      short GetID
00011F18 E2F9                <1>        loop    WaitID
00011F1A EB0F                <1>        jmp     short NextPort
                             <1> GetID:
00011F1C 6683EA04            <1>        sub     dx, 04h
00011F20 EC                  <1>        in     al, dx
00011F21 3CAA                <1>        cmp     al, 0AAh
00011F23 7413                <1>        je      short Found
00011F25 6683C204            <1>        add     dx, 04h
00011F29 E2E8                <1>        loop    WaitID
                             <1> NextPort:
00011F2B 6683C310            <1>        add     bx, 10h            ; if not response,
00011F2F 6681FB6002          <1>        cmp     bx, 260h    ; try the next port.
00011F34 76C4                <1>        jbe     short ResetDSP
00011F36 F9                  <1>        stc
00011F37 C3                  <1>        retn
                             <1> Found:
00011F38 66891D[C26B0100]    <1>        mov     [audio_io_base], bx      ; SB Port Address Found!
                             <1> ScanIRQ:
                             <1> SetIrqs:
00011F3F 28C0                <1>        sub    al, al ; 0
00011F41 A2[BA6B0100]        <1>        mov    [IRQnum], al ; reset
00011F46 A2[BF6B0100]        <1>        mov    [audio_intr], al ; reset
                             <1> 
                             <1>        ; ah > 0 -> set IRQ vector
                             <1>        ; al = IRQ number
                             <1>        ;mov   ax, 103h ; IRQ 3
                             <1>        ;call  set_hardware_int_vector
                             <1>        ;mov   ax, 104h ; IRQ 4
                             <1>        ;call  set_hardware_int_vector
00011F4B 66B80501            <1>        mov    ax, 105h ; IRQ 5
00011F4F E8F0DDFFFF          <1>        call   set_hardware_int_vector
00011F54 66B80701            <1>        mov    ax, 107h ; IRQ 7
00011F58 E8E7DDFFFF          <1>        call   set_hardware_int_vector
                             <1> 
00011F5D 668B15[C26B0100]    <1>        mov     dx, [audio_io_base] ; tells to the SB to
00011F64 6683C20C            <1>        add     dx, 0Ch                ; generate a IRQ!
                             <1> WaitSb:
00011F68 EC                  <1>        in     al, dx
00011F69 08C0                <1>        or      al, al
00011F6B 78FB                <1>        js      short WaitSb
00011F6D B0F2                <1>        mov     al, 0F2h
00011F6F EE                  <1>        out    dx, al
                             <1> 
00011F70 31C9                <1>        xor     ecx, ecx    ; wait until IRQ level
                             <1> WaitIRQ:
00011F72 A0[BA6B0100]        <1>        mov    al, [IRQnum]
00011F77 3C00                <1>        cmp     al, 0 ; is changed or timeout.
00011F79 7706                <1>        ja     short IrqOk
00011F7B 6649                <1>        dec    cx
00011F7D 75F3                <1>        jnz    short WaitIRQ
00011F7F EB15                <1>        jmp    short RestoreIrqs
                             <1> IrqOk:
00011F81 A2[BF6B0100]        <1>        mov    [audio_intr], al ; set
00011F86 668B15[C26B0100]    <1>        mov     dx, [audio_io_base]
00011F8D 6683C20E            <1>        add     dx, 0Eh
00011F91 EC                  <1>        in     al, dx ; SB acknowledge.
00011F92 B020                <1>        mov     al, 20h
00011F94 E620                <1>        out    20h, al      ; Hardware acknowledge.
                             <1> 
                             <1> RestoreIrqs:
                             <1>        ; ah = 0 -> reset IRQ vector
                             <1>        ; al = IRQ number
                             <1>        ;mov   ax, 3 ; IRQ 3
                             <1>        ;call  set_hardware_int_vector
                             <1>        ;mov   ax, 4 ; IRQ 4
                             <1>        ;call  set_hardware_int_vector
00011F96 66B80500            <1>        mov    ax, 5 ; IRQ 5
00011F9A E8A5DDFFFF          <1>        call   set_hardware_int_vector
00011F9F 66B80700            <1>        mov    ax, 7 ; IRQ 7
00011FA3 E89CDDFFFF          <1>        call   set_hardware_int_vector
                             <1> 
00011FA8 31D2                <1>        xor    edx, edx
00011FAA 8915[C46B0100]      <1>        mov    [audio_dev_id], edx ; 0
00011FB0 8915[C86B0100]      <1>        mov    [audio_vendor], edx ; 0
00011FB6 8915[CC6B0100]      <1>        mov    [audio_stats_cmd], edx ; 0
                             <1> 
                             <1>        ;popad
                             <1> 
00011FBC 803D[BF6B0100]01    <1>        cmp     byte [audio_intr], 1 ; IRQ level was changed?
                             <1> 
00011FC3 C3                  <1>        retn
                             <1> 
                             <1> %macro       SbOut  1
                             <1> %%Wait:
                             <1>        in     al, dx
                             <1>        or     al, al
                             <1>        js     short %%Wait
                             <1>        mov    al, %1
                             <1>        out    dx, al
                             <1> %endmacro
                             <1> 
                             <1> SbInit_play:
                             <1>        ; 22/10/2017
                             <1>        ; 20/10/2017
```

```
                              <1>       ; 06/10/2017
                              <1>       ; 13/07/2017, 09/08/2017
                              <1>       ; 24/04/2017, 15/05/2017, 24/06/2017
                              <1>       ;pushad
                              <1> SetBuffer:
                              <1>       ;mov   byte [DmaFlag], 0
                              <1>
00011FC4 8B1D[DC6B0100]       <1>       mov    ebx, [audio_dma_buff] ; physical addr of DMA buff
00011FCA 89DF                 <1>       mov    edi, ebx
00011FCC 8B0D[E06B0100]       <1>       mov      ecx, [audio_dmabuff_size]
                              <1>
00011FD2 803D[EC6B0100]10     <1>       cmp    byte [audio_bps], 16
00011FD9 7531                 <1>       jne    short sbInit_0 ; set 8 bit DMA buffer
                              <1>
                              <1>       ; 09/08/2017
                              <1>       ; convert byte count to word count
00011FDB D1E9                 <1>       shr    ecx, 1
00011FDD 49                   <1>       dec    ecx ; word count - 1
                              <1>       ; convert byte offset to word offset
00011FDE D1EB                 <1>       shr    ebx, 1
                              <1>
                              <1>       ; 16 bit DMA buffer setting (DMA channel 5)
00011FE0 B005                 <1>       mov     al, 05h  ; set mask bit for channel 5  (4+1)
00011FE2 E6D4                 <1>       out    0D4h, al
                              <1>
00011FE4 30C0                 <1>       xor     al, al   ; stops all DMA processes on selected channel
00011FE6 E6D8                 <1>       out    0D8h, al ; clear selected channel register
                              <1>
00011FE8 88D8                 <1>       mov     al, bl       ; byte 0 of DMA buffer offset in words (physical)
00011FEA E6C4                 <1>       out    0C4h, al ; DMA channel 5 port number
                              <1>
00011FEC 88F8                 <1>       mov     al, bh   ; byte 1 of DMA buffer offset in words (physical)
00011FEE E6C4                 <1>       out    0C4h, al
                              <1>
                              <1>       ; 09/08/2017
00011FF0 C1EB0F               <1>       shr    ebx, 15       ; complete 16 bit shift
00011FF3 80E3FE               <1>       and    bl, 0FEh ; clear bit 0 (not necessary, it will be ignored)
                              <1>
00011FF6 88D8                 <1>       mov     al, bl   ; byte 2 of DMA buffer address (physical)
00011FF8 E68B                 <1>       out    8Bh, al  ; page register port addr for channel 5 ; 13/07/2017
                              <1>
00011FFA 88C8                 <1>       mov     al, cl   ; low byte of DMA count - 1
00011FFC E6C6                 <1>       out    0C6h, al ; count register port addr for channel 1
                              <1>
00011FFE 88E8                 <1>       mov     al, ch   ; high byte of DMA count - 1
00012000 E6C6                 <1>       out    0C6h, al
                              <1>
                              <1>       ; channel 5, read, autoinitialized, single mode
                              <1>       ;mov   al, 49h
00012002 B059                 <1>       mov    al, 59h  ; 06/10/2017
00012004 E6D6                 <1>       out    0D6h, al ; DMA mode register port address
                              <1>
00012006 B001                 <1>       mov     al, 01h  ; clear mask bit for channel 1
00012008 E6D4                 <1>       out    0D4h, al ; DMA mask register port address
                              <1>
0001200A EB28                 <1>       jmp    short ClearBuffer
                              <1>
                              <1> sbInit_0:
0001200C 49                   <1>       dec     ecx ; 09/08/2017
                              <1>
                              <1>       ; 8 bit DMA buffer setting (DMA channel 1)
0001200D B005                 <1>       mov     al, 05h ; set mask bit for channel 1  (4+1)
0001200F E60A                 <1>       out    0Ah, al ; DMA mask register
                              <1>
00012011 30C0                 <1>       xor     al, al  ; stops all DMA processes on selected channel
00012013 E60C                 <1>       out    0Ch, al ; clear selected channel register
                              <1>
00012015 88D8                 <1>       mov     al, bl      ; byte 0 of DMA buffer address (physical)
00012017 E602                 <1>       out    02h, al ; DMA channel 1 port number
                              <1>
00012019 88F8                 <1>       mov     al, bh  ; byte 1 of DMA buffer address (physical)
0001201B E602                 <1>       out    02h, al
                              <1>
0001201D C1EB10               <1>       shr    ebx, 16
                              <1>
00012020 88D8                 <1>       mov     al, bl  ; byte 2 of DMA buffer address (physical)
00012022 E683                 <1>       out    83h, al ; page register port addr for channel 1
                              <1>
00012024 88C8                 <1>       mov     al, cl  ; low byte of DMA count - 1
00012026 E603                 <1>       out    03h, al ; count register port addr for channel 1
                              <1>
00012028 88E8                 <1>       mov     al, ch  ; high byte of DMA count - 1
0001202A E603                 <1>       out    03h, al
                              <1>
                              <1>       ; channel 1, read, autoinitialized, single mode
                              <1>       ;mov   al, 49h
0001202C B059                 <1>       mov    al, 59h ; 06/10/2017
0001202E E60B                 <1>       out    0Bh, al ; DMA mode register port address
                              <1>
00012030 B001                 <1>       mov     al, 01h ; clear mask bit for channel 1
00012032 E60A                 <1>       out    0Ah, al ; DMA mask register port address
                              <1>
                              <1> ClearBuffer:
                              <1>       ;;mov  edi, [audio_dma_buff]
                              <1>       ;;mov  ecx, [audio_dmabuff_size]
                              <1>       ;inc   ecx
                              <1>       ;mov     al, 80h
                              <1>       ;;cld
                              <1>       ;rep     stosb
                              <1> SetIrq:
                              <1>       ;mov   ebx, SbIrqhandler
                              <1>       ;mov   al, [audio_intr] ; IRQ number
                              <1>       ;call  set_dev_IRQ_service
```

```
                             <1>       ;; SETUP (audio) INTERRUPT CALLBACK SERVICE
                             <1>       ;mov   bl, [audio_intr] ; IRQ number
                             <1>       ;mov   bh, [audio_cb_mode]
                             <1>       ;inc   bh  ; 1 = Signal Response Byte method (fixed value)
                             <1>       ;          ; 2 = Callback service method
                             <1>       ;          ; 3 = Auto Increment S.R.B. method
                             <1>       ;mov   cl, [audio_srb]
                             <1>       ;mov   edx, [audio_cb_addr]
                             <1>       ;mov   al, [audio_user]
                             <1>       ;call  set_irq_callback_service
                             <1> ResetDsp:
00012034 668B15[C26B0100]    <1>       mov     dx, [audio_io_base]
0001203B 6683C206            <1>       add     dx, 06h
0001203F B001                <1>       mov     al, 1
00012041 EE                  <1>       out    dx, al
                             <1>
00012042 EC                  <1>       in     al, dx
00012043 EC                  <1>       in     al, dx
00012044 EC                  <1>       in     al, dx
00012045 EC                  <1>       in     al, dx
                             <1>
00012046 30C0                <1>       xor     al, al
00012048 EE                  <1>       out    dx, al
                             <1>
00012049 66B96400            <1>       mov     cx, 100
0001204D 28E4                <1>       sub     ah, ah ; 0
                             <1> WaitId:
0001204F 668B15[C26B0100]    <1>       mov     dx, [audio_io_base]
00012056 6683C20E            <1>       add     dx, 0Eh
0001205A EC                  <1>       in     al, dx
0001205B 08C0                <1>       or      al, al
0001205D 7807                <1>       js      short sb_GetId
0001205F E2EE                <1>       loop    WaitId
00012061 E9B4000000          <1>       jmp     sb_Exit
                             <1> sb_GetId:
00012066 668B15[C26B0100]    <1>       mov     dx, [audio_io_base]
0001206D 6683C20A            <1>       add     dx, 0Ah
00012071 EC                  <1>       in     al, dx
00012072 3CAA                <1>       cmp     al, 0AAh
00012074 7407                <1>       je      short SbOk
00012076 E2D7                <1>       loop    WaitId
00012078 E99D000000          <1>       jmp    sb_Exit
                             <1> SbOk:
0001207D 668B15[C26B0100]    <1>       mov     dx, [audio_io_base]
00012084 6683C20C            <1>       add     dx, 0Ch
                             <1>       SbOut   0D1h ; Turn on speaker
                             <2> %%Wait:
00012088 EC                  <2>  in al, dx
00012089 08C0                <2>  or al, al
0001208B 78FB                <2>  js short %%Wait
0001208D B0D1                <2>  mov al, %1
0001208F EE                  <2>  out dx, al
                             <1>       SbOut   41h ; 8 bit or 16 bit transfer
                             <2> %%Wait:
00012090 EC                  <2>  in al, dx
00012091 08C0                <2>  or al, al
00012093 78FB                <2>  js short %%Wait
00012095 B041                <2>  mov al, %1
00012097 EE                  <2>  out dx, al
00012098 668B1D[EE6B0100]    <1>       mov    bx, [audio_freq] ; sampling rate (Hz)
                             <1>       SbOut  bh ; sampling rate high byte
                             <2> %%Wait:
0001209F EC                  <2>  in al, dx
000120A0 08C0                <2>  or al, al
000120A2 78FB                <2>  js short %%Wait
000120A4 88F8                <2>  mov al, %1
000120A6 EE                  <2>  out dx, al
                             <1>       SbOut  bl ; sampling rate low byte
                             <2> %%Wait:
000120A7 EC                  <2>  in al, dx
000120A8 08C0                <2>  or al, al
000120AA 78FB                <2>  js short %%Wait
000120AC 88D8                <2>  mov al, %1
000120AE EE                  <2>  out dx, al
                             <1>
                             <1>       ; 22/05/2017
000120AF E8C0000000          <1>       call   sb16_volume_initial ; 15/05/2017
                             <1>       ; 20/05/2017
                             <1>       ;call  sb16_volume
                             <1>
                             <1> StartDma:
                             <1>       ; autoinitialized mode
000120B4 803D[EC6B0100]10    <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
000120BB 7411                <1>       je     short sb_play_1
                             <1>       ; 8 bit samples
000120BD 66BBC600            <1>       mov    bx, 0C6h ; 8 bit output (0C6h)
000120C1 803D[ED6B0100]02    <1>       cmp    byte [audio_stmo], 2 ; 1 = mono, 2 = stereo
000120C8 7214                <1>       jb     short sb_play_2
000120CA B720                <1>       mov    bh, 20h       ; 8 bit stereo (20h)
000120CC EB10                <1>       jmp    short sb_play_2
                             <1> sb_play_1:
                             <1>       ; 16 bit samples
000120CE 66BBB610            <1>       mov    bx, 10B6h ; 16 bit output (0B6h)
000120D2 803D[ED6B0100]02    <1>       cmp    byte [audio_stmo], 2 ; 1 = mono, 2 = stereo
000120D9 7203                <1>       jb     short sb_play_2
000120DB 80C720              <1>       add    bh, 20h      ; 16 bit stereo (30h)
                             <1> sb_play_2:
                             <1>       ; PCM output (8/16 bit mono autoinitialized transfer)
                             <1>       SbOut   bl ; bCommand
                             <2> %%Wait:
000120DE EC                  <2>  in al, dx
000120DF 08C0                <2>  or al, al
000120E1 78FB                <2>  js short %%Wait
```

```
000120E3 88D8                <2>  mov al, %1
000120E5 EE                  <2>  out dx, al
                             <1>       SbOut  bh ; bMode
                             <2> %%Wait:
000120E6 EC                  <2>  in al, dx
000120E7 08C0                <2>  or al, al
000120E9 78FB                <2>  js short %%Wait
000120EB 88F8                <2>  mov al, %1
000120ED EE                  <2>  out dx, al
000120EE 8B1D[E06B0100]      <1>       mov    ebx, [audio_dmabuff_size]  ; 15/05/2017
000120F4 D1EB                <1>       shr    ebx, 1 ; half buffer size
                             <1>       ; 20/10/2017
000120F6 803D[EC6B0100]10    <1>       cmp    byte [audio_bps], 16 ; 16 bit DMA
000120FD 7502                <1>       jne    short sb_play_3
000120FF D1EB                <1>       shr    ebx, 1 ; byte count to word count
                             <1> sb_play_3:
00012101 664B                <1>       dec    bx  ; wBlkSize is one less than the actual size
                             <1>       SbOut   bl
                             <2> %%Wait:
00012103 EC                  <2>  in al, dx
00012104 08C0                <2>  or al, al
00012106 78FB                <2>  js short %%Wait
00012108 88D8                <2>  mov al, %1
0001210A EE                  <2>  out dx, al
                             <1>       SbOut   bh
                             <2> %%Wait:
0001210B EC                  <2>  in al, dx
0001210C 08C0                <2>  or al, al
0001210E 78FB                <2>  js short %%Wait
00012110 88F8                <2>  mov al, %1
00012112 EE                  <2>  out dx, al
                             <1>
00012113 C605[F06B0100]01    <1>       mov    byte [audio_play_cmd], 1 ; playing !
                             <1>
                             <1>       ;; Set Voice and master volumes
                             <1>       ;mov   dx, [audio_io_base]
                             <1>       ;add   dl, 4 ; Mixer chip Register Address Port
                             <1>       ;SbOut 30h   ; select Master Volume Register (L)
                             <1>       ;inc   dl    ; Mixer chip Register Data Port
                             <1>       ;SbOut 0F8h  ; Max. volume value is 31 (31*8)
                             <1>       ;dec   dl
                             <1>       ;SbOut 31h   ; select Master Volume Register (R)
                             <1>       ;inc   dl
                             <1>       ;SbOut 0F8h  ; Max. volume value is 31 (31*8)
                             <1>       ;dec   dl
                             <1>       ;SbOut 32h   ; select Voice Volume Register (L)
                             <1>       ;inc   dl
                             <1>       ;SbOut 0F8h  ; Max. volume value is 31 (31*8)
                             <1>       ;dec   dl
                             <1>       ;SbOut 33h   ; select Voice Volume Register (R)
                             <1>       ;inc   dl
                             <1>       ;SbOut 0F8h  ; Max. volume value is 31 (31*8)
                             <1>       ;;
                             <1>       ;dec   dl
                             <1>       ;SbOut 44h   ; select Treble Register (L)
                             <1>       ;inc   dl
                             <1>       ;SbOut 0F0h  ; Max. Treble value is 15 (15*16)
                             <1>       ;dec   dl
                             <1>       ;SbOut 45h   ; select Treble Register (R)
                             <1>       ;inc   dl
                             <1>       ;SbOut 0F0h  ; Max. Treble value is 15 (15*16)
                             <1>       ;dec   dl
                             <1>       ;SbOut 46h   ; select Bass Register (L)
                             <1>       ;inc   dl
                             <1>       ;SbOut 0F0h  ; Max. Bass value is 15 (15*16)
                             <1>       ;dec   dl
                             <1>       ;SbOut 47h   ; select Bass Register (R)
                             <1>       ;inc   dl
                             <1>       ;SbOut 0F0h  ; Max. Bass value is 15 (15*16)
                             <1>
                             <1> sb_Exit:
                             <1>       ;popad
0001211A C3                  <1>       retn
                             <1>
                             <1> sb16_int_handler:
                             <1>       ; Interrupt Handler for Sound Blaster 16 Audio Card
                             <1>       ; Note: called by 'dev_IRQ_service'
                             <1>       ; 20/10/2017
                             <1>       ; 12/10/2017
                             <1>       ; 10/10/2017
                             <1>       ; 12/05/2017, 09/10/2017
                             <1>       ; 24/04/2017 (TRDOS 386 kernel, 'audio.s')
                             <1>       ; 10/03/2017 - 'PLAYWAV.PRG' ('playwav.s')
                             <1>
                             <1>       ;push  eax ; * must be saved !
                             <1>       ;push  ebx ; * must be saved !
                             <1>       ;push  ecx
                             <1>       ;push  edx
                             <1>       ;push  esi
                             <1>       ;push  edi
                             <1>
0001211B 668B15[C26B0100]    <1>       mov     dx, [audio_io_base]
                             <1>       ; 20/10/2017
00012122 80C20F              <1>       add     dl, 0Fh ; 2xFh (DSP 16 bit intr ack)
00012125 803D[EC6B0100]10    <1>       cmp    byte [audio_bps], 16
0001212C 7402                <1>       je     short sb_irq_16bit_ack
                             <1> sb_irq_8bit_ack:
0001212E FECA                <1>       dec    dl  ; 2xEh (DSP 8 bit intr ack)
                             <1> sb_irq_16bit_ack:
00012130 EC                  <1>       in     al, dx
                             <1>
                             <1>       ;cmp   byte [audio_busy], 0
                             <1>       ;ja    short sb_irq_h3
                             <1>
                             <1>       ;mov   byte [audio_busy], 1
                             <1>
00012131 803D[F06B0100]01    <1>       cmp    byte [audio_play_cmd], 1
00012138 7307                <1>       jnb    short sb_irq_h1
                             <1> sb_irq_h0:
0001213A E8A9000000          <1>       call   sb16_stop
0001213F EB2B                <1>       jmp    short sb_irq_h3
                             <1> sb_irq_h1:
                             <1>       ;call  sb16_tuneloop
                             <1>       ; 09/10/2017
                             <1> sb16_tuneloop:
00012141 8B3D[DC6B0100]      <1>       mov    edi, [audio_dma_buff]
00012147 8B0D[E06B0100]      <1>       mov    ecx, [audio_dmabuff_size]
0001214D D1E9                <1>       shr    ecx, 1 ; dma buff size / 2 = half buffer size
                             <1>
                             <1>       ; 22/05/2017
0001214F F605[E46B0100]01    <1>       test   byte [audio_flag], 1  ; Current flag value
00012156 7402                <1>       jz     short sb_tlp1 ; EOL (Half Buffer 1 must be filled)
                             <1>       ; FLAG (Half Buffer 2 must be filled)
00012158 01CF                <1>       add    edi, ecx
                             <1>       ; 15/05/2017
                             <1> sb_tlp1:
0001215A 8B35[D46B0100]      <1>       mov    esi, [audio_p_buffer] ; phy addr of audio buff
                             <1>       ;rep   movsb
00012160 C1E902              <1>       shr    ecx, 2 ; half buff size / 4
00012163 F3A5                <1>       rep    movsd
                             <1>       ;retn
                             <1>
                             <1>       ; 10/10/2017
                             <1>       ; switch flag value
00012165 8035[E46B0100]01    <1>       xor    byte [audio_flag], 1
                             <1>
                             <1>       ; 12/10/2017
                             <1>       ; [audio_flag] = 0 : Playing dma half buffer 2 (odd intr count)
                             <1>                       ; Next buffer (to update) is dma half buff 1
                             <1>       ;             = 1 : Playing dma half buffer 1 (even intr count)
                             <1>                       ; Next buffer (to update) is dma half buff 2
                             <1>
                             <1> sb_irq_h3:
                             <1>       ;mov   byte [audio_busy], 0
                             <1>
                             <1>       ;pop   edi
                             <1>       ;pop   esi
                             <1>       ;pop   edx
                             <1>       ;pop   ecx
                             <1>       ;pop   ebx ; * must be restored !
                             <1>       ;pop   eax ; * must be restored !
                             <1>
0001216C C3                  <1>       retn
                             <1>
                             <1> sb16_volume:
                             <1>       ; 22/10/2017
                             <1>       ; mov [audio_master_volume_l], cl
                             <1>       ; mov [audio_master_volume_h], ch
0001216D 66890D[F26B0100]    <1>       mov    [audio_master_volume], cx
                             <1> sb16_volume_initial:
00012174 6652                <1>       push   dx ; DX (port address) must be saved
00012176 668B15[C26B0100]    <1>       mov    dx, [audio_io_base]
0001217D 6683C204            <1>       add    dx, 4 ; Mixer chip address port
00012181 B022                <1>       mov    al, 22h ; master volume
00012183 EE                  <1>       out    dx, al
00012184 6642                <1>       inc    dx
00012186 8A25[F26B0100]      <1>       mov    ah, [audio_master_volume_l]
0001218C C0EC02              <1>       shr    ah, 2 ; 32 -> 8 level
0001218F C0E405              <1>       shl    ah, 5 ; bit 5 to 7
00012192 A0[F36B0100]        <1>       mov    al, [audio_master_volume_r]
00012197 C0E802              <1>       shr    al, 2 ; 32 -> 8 level
                             <1>       ;and   al, 0Fh
0001219A D0E0                <1>       shl    al, 1 ; bit 1 to 3
0001219C 08E0                <1>       or     al, ah
0001219E EE                  <1>       out    dx, al
0001219F 665A                <1>       pop    dx ; DX (port address) must be restored
000121A1 C3                  <1>       retn
                             <1>
                             <1> sb16_pause:
000121A2 668B15[C26B0100]    <1>       mov    dx, [audio_io_base]
000121A9 6683C20C            <1>       add    dx, 0Ch ; Command & Data Port
000121AD 803D[EC6B0100]10    <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
000121B4 7404                <1>       je     short sb_pause_1
                             <1>       ; 8 bit samples
000121B6 B3D0                <1>       mov    bl, 0D0h ; 8 bit DMA mode
000121B8 EB02                <1>       jmp    short sb_pause_2
                             <1> sb_pause_1:
                             <1>       ; 16 bit samples
000121BA B3D5                <1>       mov    bl, 0D5h ; 16 bit DMA mode
                             <1> sb_pause_2:
                             <1>       SbOut   bl ; bCommand
                             <2> %%Wait:
000121BC EC                  <2>  in al, dx
000121BD 08C0                <2>  or al, al
000121BF 78FB                <2>  js short %%Wait
000121C1 88D8                <2>  mov al, %1
000121C3 EE                  <2>  out dx, al
                             <1> sb_pause_3:
000121C4 C3                  <1>       retn
                             <1>
                             <1> sb16_continue:
000121C5 668B15[C26B0100]    <1>       mov    dx, [audio_io_base]
000121CC 6683C20C            <1>       add    dx, 0Ch ; Command & Data Port
000121D0 803D[EC6B0100]10    <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
000121D7 7404                <1>       je     short sb_cont_1
                             <1>       ; 8 bit samples
```

```
000121D9 B3D4                <1>       mov    bl, 0D4h ; 8 bit DMA mode
000121DB EB02                <1>       jmp    short sb_cont_2
                             <1> sb_cont_1:
                             <1>       ; 16 bit samples
000121DD B3D6                <1>       mov    bl, 0D6h ; 16 bit DMA mode
                             <1> sb_cont_2:
                             <1>       SbOut   bl ; bCommand
                             <2> %%Wait:
000121DF EC                  <2>  in al, dx
000121E0 08C0                <2>  or al, al
000121E2 78FB                <2>  js short %%Wait
000121E4 88D8                <2>  mov al, %1
000121E6 EE                  <2>  out dx, al
                             <1> sb_cont_3:
000121E7 C3                  <1>       retn
                             <1>
                             <1> sb16_stop:
                             <1>       ; 24/04/2017
000121E8 803D[F06B0100]00    <1>       cmp    byte [audio_play_cmd], 0
000121EF 7648                <1>       jna    short sb16_stop_4
                             <1>
                             <1>       ; 22/05/2017
000121F1 668B15[C26B0100]    <1>       mov    dx, [audio_io_base]
000121F8 6683C20C            <1>       add    dx, 0Ch
                             <1>
000121FC B3D9                <1>       mov    bl, 0D9h ; exit auto-initialize 16 bit transfer
                             <1>       ; stop  autoinitialized DMA transfer mode
000121FE 803D[EC6B0100]10    <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
00012205 7402                <1>       je     short sb16_stop_1
                             <1>       ;mov   bl, 0DAh ; exit auto-initialize 8 bit transfer
00012207 FEC3                <1>       inc    bl
                             <1> sb16_stop_1:
                             <1>       SbOut bl ; exit auto-initialize transfer command
                             <2> %%Wait:
00012209 EC                  <2>  in al, dx
0001220A 08C0                <2>  or al, al
0001220C 78FB                <2>  js short %%Wait
0001220E 88D8                <2>  mov al, %1
00012210 EE                  <2>  out dx, al
                             <1>
00012211 30C0                <1>       xor     al, al ; stops all DMA processes on selected channel
                             <1>
00012213 803D[EC6B0100]10    <1>       cmp    byte [audio_bps], 16 ; 16 bit samples
0001221A 7404                <1>       je     short sb16_stop_2
0001221C E60C                <1>       out    0Ch, al ; clear selected channel register
0001221E EB02                <1>       jmp    short sb16_stop_3
                             <1>
                             <1> sb16_stop_2:
00012220 E6D8                <1>       out    0D8h, al ; clear selected channel register
                             <1>
                             <1> sb16_stop_3:
00012222 C605[F06B0100]00    <1>       mov    byte [audio_play_cmd], 0 ; stop !
                             <1> SbDone:
                             <1>       ;mov   dx, [audio_io_base]
                             <1>       ;add   dx, 0Ch
                             <1>       SbOut   0D0h
                             <2> %%Wait:
00012229 EC                  <2>  in al, dx
0001222A 08C0                <2>  or al, al
0001222C 78FB                <2>  js short %%Wait
0001222E B0D0                <2>  mov al, %1
00012230 EE                  <2>  out dx, al
                             <1>       SbOut   0D3h
                             <2> %%Wait:
00012231 EC                  <2>  in al, dx
00012232 08C0                <2>  or al, al
00012234 78FB                <2>  js short %%Wait
00012236 B0D3                <2>  mov al, %1
00012238 EE                  <2>  out dx, al
                             <1> sb16_stop_4:
00012239 C3                  <1>       retn
                             <1>
                             <1> sb16_reset:
                             <1>       ; 24/04/2017
0001223A 668B15[C26B0100]    <1>       mov     dx, [audio_io_base] ; try to reset the DSP.
00012241 6683C206            <1>       add     dx, 06h
00012245 B001                <1>       mov    al, 1
00012247 EE                  <1>       out    dx, al
                             <1>
00012248 EC                  <1>       in     al, dx
00012249 EC                  <1>       in     al, dx
0001224A EC                  <1>       in     al, dx
0001224B EC                  <1>       in     al, dx
                             <1>
0001224C 30C0                <1>       xor     al, al
0001224E EE                  <1>       out    dx, al
                             <1>
0001224F 6683C208            <1>       add     dx, 08h
00012253 66B96400            <1>       mov    cx, 100
                             <1> sbrstWaitID:
00012257 EC                  <1>       in     al, dx
00012258 08C0                <1>       or      al, al
0001225A 7804                <1>       js      short sbrstGetID
0001225C E2F9                <1>       loop    sbrstWaitID
0001225E F9                  <1>       stc
0001225F C3                  <1>       retn
                             <1> sbrstGetID:
00012260 6683EA04            <1>       sub     dx, 04h
00012264 EC                  <1>       in     al, dx
00012265 3CAA                <1>       cmp     al, 0AAh
00012267 7406                <1>       je      short sb_rst_retn
00012269 6683C204            <1>       add     dx, 04h
0001226D E2E8                <1>       loop    sbrstWaitID
```

```
                             <1> sb_rst_retn:
0001226F C3                  <1>       retn
                             <1>
                             <1> ac97_codec_config:
                             <1>       ; 10/06/2017
                             <1>       ; 05/06/2017
                             <1>       ; 29/05/2017
                             <1>       ; 28/05/2017 (TRDOS 386, 'audio.s')
                             <1>       ; 07/11/2016 (Erdogan Tan)
                             <1>       ; Derived from 'codecConfig' procedure in 'CODEC.ASM'
                             <1>       ; .wav player for DOS by Jeff Leyda (02/09/2002)
                             <1>
                             <1>       ;; 'PLAYER.ASM'
                             <1>       ;; get ICH base address regs for mixer and bus master
                             <1>
                             <1> init_ac97_controller: ; 10/06/2017
00012270 A1[C46B0100]        <1>       mov    eax, [audio_dev_id]
                             <1>         ;mov al, NAMBAR_REG
                             <1>         ;;call  pciRegRead16                  ; read PCI registers 10-11
                             <1>         ;call      pciRegRead32
                             <1>       ;and   dx, IO_ADDR_MASK          ; mask off BIT0
                             <1>       ;;and  edx, IO_ADDR_MASK
                             <1>
                             <1>         ;mov [NAMBAR], dx              ; save audio mixer base addr
                             <1>
                             <1>         ;mov    al, NABMBAR_REG
                             <1>         ;;call     pciRegRead16
                             <1>         ;call      pciRegRead32
                             <1>       ;and   dx, 0FFC0h ; IO_ADDR_MASK
                             <1>       ;;and  edx, 0FFC0h
                             <1>
                             <1>         ;mov    [NABMBAR], dx                 ; save bus master base addr
                             <1>
                             <1>       ;mov   eax, [audio_dev_id]
00012275 B004                <1>         mov     al, PCI_CMD_REG
                             <1>         ;call      pciRegRead8                ; read PCI command register
00012277 E840F8FFFF          <1>         call pciRegRead16
0001227C 80CA05              <1>       or      dl, IO_ENA+BM_ENA                ; enable IO and bus master
                             <1>         ;call      pciRegWrite8
0001227F E8A3F8FFFF          <1>       call   pciRegWrite16
                             <1>
                             <1>       ; 'CODEC.ASM'
                             <1>
                             <1>       ; enable codec, unmute stuff, set output rate
                             <1> ;     ; entry: [audio_freq] = desired sample rate
                             <1>
                             <1> ;     mov          dx, [NAMBAR]
                             <1> ;     add          dx, CODEC_EXT_AUDIO_CTRL_REG     ; 2Ah
                             <1> ;     in           ax, dx
                             <1> ;     or     ax, 1
                             <1> ;     out    dx, ax                           ; Enable variable rate audio
                             <1>
                             <1> ;       ;call    delay1_4ms
                             <1> ;       ;call    delay1_4ms
                             <1> ;       ;call    delay1_4ms
                             <1> ;       ;call    delay1_4ms
                             <1>
                             <1> ;     mov    ax, [audio_freq]          ; sample rate
                             <1>
                             <1> ;     mov          dx, [NAMBAR]
                             <1> ;     add          dx, CODEC_PCM_FRONT_DACRATE_REG  ; 2Ch
                             <1> ;     out    dx, ax                           ; out sample rate
                             <1>
                             <1> ;       ;call      delay1_4ms
                             <1> ;       ;call      delay1_4ms
                             <1> ;       ;call      delay1_4ms
                             <1> ;       ;call      delay1_4ms
                             <1>
                             <1>       ;mov   dx, [NAMBAR]              ; mixer base address
                             <1>         ;add dx, CODEC_RESET_REG              ; reset register
                             <1>         ;mov ax, 42
                             <1>       ;out   dx, ax                          ; reset
                             <1>
                             <1>       ;mov    dx, [NABMBAR]                  ; bus master base address
                             <1>         ;add dx, GLOB_STS_REG
                             <1>         ;mov ax, 2
                             <1>       ;out   dx, ax
                             <1>
00012284 E831F9FFFF          <1>         call    delay_100ms ; 29/05/2017
                             <1>
                             <1> init_ac97_codec:
                             <1>       ; 10/06/2017
                             <1>       ; 29/05/2017
                             <1>       ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, intelac97.asm)
                             <1>       ;
00012289 66BA2C00            <1>       mov    dx, GLOB_CNT_REG ; 2Ch
0001228D 660315[F66B0100]    <1>       add    dx, [NABMBAR]
00012294 ED                  <1>       in     eax, dx
                             <1>       ; ?
00012295 66BA3000            <1>       mov    dx, GLOB_STS_REG ; 30h
00012299 660315[F66B0100]    <1>       add    dx, [NABMBAR]
000122A0 ED                  <1>       in     eax, dx
                             <1>
000122A1 83F8FF              <1>       cmp    eax, 0FFFFFFFFh ; -1
000122A4 744B                <1>       je     short init_ac97_codec_err1
                             <1>
000122A6 A900030010          <1>       test   eax, CTRL_ST_CREADY
000122AB 7507                <1>       jnz    short _ac97_codec_ready
                             <1>
000122AD E8EF020000          <1>       call   reset_ac97_codec
000122B2 723E                <1>       jc     short init_ac97_codec_err2
                             <1>
                             <1> _ac97_codec_ready:
```

```
000122B4 668B15[F46B0100]    <1>       mov    dx, [NAMBAR]
                             <1>       ;add   dx, 0 ; ac_reg_0 ; reset register
000122BB 66EF                <1>       out    dx, ax
                             <1> 
000122BD 31C0                <1>       xor    eax, eax ; 0
000122BF 668B15[F46B0100]    <1>       mov    dx, [NAMBAR]
000122C6 6683C226            <1>       add    dx, CODEC_REG_POWERDOWN
000122CA 66EF                <1>       out    dx, ax
                             <1> 
                             <1>       ; 10/06/2017
                             <1>       ; 29/05/2017
                             <1>       ; wait for 1 second
000122CC B9E8030000          <1>       mov    ecx, 1000 ; 1000*0.25ms = 1s
                             <1> _ac97_codec_rloop:
000122D1 E8F1F8FFFF          <1>       call   delay1_4ms
000122D6 E8ECF8FFFF          <1>       call   delay1_4ms
000122DB E8E7F8FFFF          <1>       call   delay1_4ms
000122E0 E8E2F8FFFF          <1>       call   delay1_4ms
                             <1>       ;mov   dx, [NAMBAR]
                             <1>       ;add   dx, CODEC_REG_POWERDOWN
000122E5 66ED                <1>       in     ax, dx
000122E7 6683E00F            <1>       and    ax, 0Fh
000122EB 3C0F                <1>       cmp    al, 0Fh
000122ED 7404                <1>       je     short _ac97_codec_init_ok
000122EF E2E0                <1>       loop   _ac97_codec_rloop
                             <1> 
                             <1> init_ac97_codec_err1:
000122F1 F9                  <1>       stc
                             <1> init_ac97_codec_err2:
000122F2 C3                  <1>       retn
                             <1> 
                             <1> _ac97_codec_init_ok:
000122F3 B002                <1>         mov     al, 2 ; force set 16-bit 2-channel PCM
000122F5 66BA2C00            <1>       mov    dx, GLOB_CNT_REG ; 2Ch
000122F9 660315[F66B0100]    <1>       add    dx, [NABMBAR]
00012300 EF                  <1>       out    dx, eax
                             <1> 
                             <1>       ;call  delay1_4ms
                             <1> 
                             <1>       ; 10/06/2017
00012301 E849020000          <1>       call   reset_ac97_controller
                             <1> 
                             <1> ;     call   setup_ac97_codec
                             <1> ;
                             <1> ;detect_ac97_codec:
                             <1> ;     retn
                             <1> 
                             <1> setup_ac97_codec:
                             <1>       ; 10/06/2017
                             <1>       ; 29/05/2017
00012306 B802020000          <1>       mov     eax, 0202h
0001230B 66A3[F26B0100]      <1>       mov    [audio_master_volume], ax
00012311 66B81F1F            <1>       mov    ax, 1F1Fh ; 31, 31
                             <1> 
00012315 668B15[F46B0100]    <1>       mov     dx, [NAMBAR]
0001231C 6683C202            <1>       add     dx, CODEC_MASTER_VOL_REG        ;02h
00012320 6631C0              <1>       xor    ax, ax       ; volume attenuation = 0 (max. volume)
00012323 66EF                <1>       out     dx, ax
                             <1> 
00012325 668B15[F46B0100]    <1>       mov     dx, [NAMBAR]
0001232C 6683C206            <1>       add     dx, CODEC_MASTER_MONO_VOL_REG   ;06h
                             <1>       ;xor   ax, ax
00012330 66EF                <1>       out     dx, ax
                             <1> 
00012332 668B15[F46B0100]    <1>       mov     dx, [NAMBAR]
00012339 6683C20A            <1>       add     dx, CODEC_PCBEEP_VOL_REG        ;0Ah
                             <1>       ;xor    ax, ax
0001233D 66EF                <1>       out     dx, ax
                             <1> 
0001233F 668B15[F46B0100]    <1>       mov     dx, [NAMBAR]
00012346 6683C218            <1>       add     dx, CODEC_PCM_OUT_REG          ;18h
                             <1>       ;xor    ax, ax
0001234A 66EF                <1>       out     dx, ax
                             <1> 
0001234C 66B80880            <1>       mov     ax,  8008h ; Mute
00012350 668B15[F46B0100]    <1>       mov     dx, [NAMBAR]
00012357 6683C20C            <1>       add    dx, 0Ch       ; AC97_PHONE_VOL ; TAD Input (Mono)
0001235B 66EF                <1>       out     dx, ax
                             <1> 
0001235D 66B80808            <1>         mov ax,  0808h
00012361 668B15[F46B0100]    <1>       mov     dx, [NAMBAR]
00012368 6683C210            <1>         add dx, CODEC_LINE_IN_VOL_REG ; 10h ; Line Input (Stereo)
0001236C 66EF                <1>       out     dx, ax
                             <1> 
                             <1>       ;mov   ax,  0808h
0001236E 668B15[F46B0100]    <1>       mov     dx, [NAMBAR]
00012375 6683C212            <1>         add dx, CODEC_CD_VOL_REG ; 12h ; CR Input (Stereo)
00012379 66EF                <1>       out     dx, ax
                             <1> 
                             <1>       ;mov   ax,  0808h
0001237B 668B15[F46B0100]    <1>       mov     dx, [NAMBAR]
00012382 6683C216            <1>         add dx, CODEC_AUX_VOL_REG ; 16h ; Aux Input (Stereo)
00012386 66EF                <1>       out     dx, ax
                             <1> 
                             <1>         ;call    delay1_4ms
                             <1>         ;call    delay1_4ms
                             <1>         ;call    delay1_4ms
                             <1>         ;call    delay1_4ms
                             <1> 
                             <1> detect_ac97_codec:
00012388 C3                  <1>         retn
                             <1> 
                             <1> set_ac97_bdl: ; Set AC97 (ICH) Buffer Descriptor List
                             <1>       ; 17/06/2017
                             <1>       ; 11/06/2017
                             <1>       ; 28/05/2017
                             <1>       ; eax = dma buffer address = [audio_DMA_buff]
                             <1>       ; ecx = dma buffer buffer size = [audio_dmabuff_size]
                             <1>
00012389 D1E9                <1>       shr    ecx, 1 ; dma half buffer size
0001238B 89CE                <1>       mov    esi, ecx
                             <1>
0001238D BF[F86B0100]        <1>         mov     edi, audio_bdl_buff    ; get BDL address
00012392 B910000000          <1>         mov     ecx, 32 / 2            ; make 32 entries in BDL
                             <1>
00012397 EB05                <1>       jmp    short s_ac97_bdl1
                             <1>
                             <1> s_ac97_bdl0:
                             <1>       ; set buffer descriptor 0 to start of data file in memory
                             <1>
00012399 A1[DC6B0100]        <1>       mov    eax, [audio_dma_buff]     ; Physical address of DMA buffer
                             <1>
                             <1> s_ac97_bdl1:
0001239E AB                  <1>       stosd                     ; store dmabuffer1 address
                             <1>
0001239F 89C2                <1>       mov    edx, eax
                             <1>
                             <1> ;
                             <1> ; Buffer Descriptors List
                             <1> ; As stated earlier, each buffer descriptor list is a set of (up to) 32
                             <1> ; descriptors, each 8 bytes in length.  Bytes 0-3 of a descriptor entry point
                             <1> ; to a chunk of memory to either play from or record to.  Bytes 4-7 of an
                             <1> ; entry describe various control things detailed below.
                             <1> ;
                             <1> ; Buffer pointers must always be aligned on a Dword boundry.
                             <1> ;
                             <1> ;
                             <1>
                             <1> ;IOC                     equ     BIT31 ; Fire an interrupt whenever this
                             <1>                                        ; buffer is complete.
                             <1>
                             <1> ;BUP                     equ     BIT30  ; Buffer Underrun Policy.
                             <1>                                        ; if this buffer is the last buffer
                             <1>                                        ; in a playback, fill the remaining
                             <1>                                        ; samples with 0 (silence) or not.
                             <1>                                        ; It's a good idea to set this to 1
                             <1>                                        ; for the last buffer in playback,
                             <1>                                        ; otherwise you're likely to get a lot
                             <1>                                        ; of noise at the end of the sound.
                             <1>
                             <1> ;
                             <1> ; Bits 15:0 contain the length of the buffer, in number of samples, which
                             <1> ; are 16 bits each, coupled in left and right pairs, or 32bits each.
                             <1> ; Luckily for us, that's the same format as .wav files.
                             <1> ;
                             <1> ; A value of FFFF is 65536 samples.  Running at 44.1Khz, that's just about
                             <1> ; 1.5 seconds of sample time.  FFFF * 32bits is 1FFFFh bytes or 128k of data.
                             <1> ;
                             <1> ; A value of 0 in these bits means play no samples.
                             <1> ;
                             <1>
000123A1 89F0                <1>       mov    eax, esi ; DMA half buffer size
000123A3 01C2                <1>       add    edx, eax
000123A5 D1E8                <1>       shr    eax, 1 ; count of 16 bit samples
                             <1>       ;or    eax, IOC+BUP
000123A7 0D00000080          <1>       or     eax, IOC ; 11/06/2017
000123AC AB                  <1>       stosd
                             <1>
                             <1> ; 2nd buffer:
                             <1>
000123AD 89D0                <1>         mov  eax, edx ; Physical address of the 2nd half of DMA buffer
000123AF AB                  <1>       stosd         ; store dmabuffer2 address
                             <1>
                             <1> ; set length to [audio_dmabuff_size]/2
                             <1> ; Set control (bits 31:16) to BUP, bits 15:0=number of samples
                             <1> ;
000123B0 89F0                <1>       mov    eax, esi ; DMA half buffer size
000123B2 D1E8                <1>       shr    eax, 1 ; count of 16 bit samples
                             <1>       ;or    eax, IOC+BUP
000123B4 0D00000080          <1>       or     eax, IOC ; 11/06/2017
000123B9 AB                  <1>       stosd
                             <1>
000123BA E2DD                <1>       loop    s_ac97_bdl0
                             <1>
000123BC C3                  <1>       retn
                             <1>
                             <1> ac97_start_play:
                             <1>       ; 28/05/2017
                             <1>       ; Derived from 'playWav' procedure in 'ICHWAV.ASM'
                             <1>       ; .wav player for DOS by Jeff Leyda (02/09/2002)
                             <1>
                             <1>       ; set output rate
                             <1>       ; entry: [audio_freq] = desired sample rate
                             <1>
000123BD 668B15[F46B0100]    <1>       mov          dx, [NAMBAR]
000123C4 6683C22A            <1>       add          dx, CODEC_EXT_AUDIO_CTRL_REG     ; 2Ah
000123C8 66ED                <1>       in           ax, dx
000123CA 6683C801            <1>       or     ax, 1
000123CE 66EF                <1>       out    dx, ax                           ; Enable variable rate audio
                             <1>
                             <1>        ;call    delay1_4ms
                             <1>        ;call    delay1_4ms
                             <1>        ;call    delay1_4ms
                             <1>        ;call    delay1_4ms
                             <1>
000123D0 66A1[EE6B0100]      <1>       mov    ax, [audio_freq]          ; sample rate
```

```
                             <1>
000123D6 668B15[F46B0100]    <1>       mov          dx, [NAMBAR]
000123DD 6683C22C            <1>       add          dx, CODEC_PCM_FRONT_DACRATE_REG  ; 2Ch
000123E1 66EF                <1>       out    dx, ax                          ; out sample rate
                             <1>
                             <1>        ;call    delay1_4ms
                             <1>        ;call    delay1_4ms
                             <1>        ;call    delay1_4ms
                             <1>        ;call    delay1_4ms
                             <1>
                             <1> ;
                             <1> ; register reset the DMA engine.  This may cause a pop noise on the output
                             <1> ; lines when the device is reset.  Prolly a better idea to mute output, then
                             <1> ; reset.
                             <1> ;
000123E3 668B15[F66B0100]    <1>         mov     dx, [NABMBAR]
000123EA 6683C21B            <1>         add     dx, PO_CR_REG                  ; set pointer to Cntl reg
000123EE B002                <1>         mov     al, RR                         ; set reset
000123F0 EE                  <1>         out     dx, al                         ; self clearing bit
                             <1> ;
                             <1> ;     mov    edi, audio_bdl_buff
                             <1> ;     mov    edx, [audio_dmabuff_size]
                             <1> ;     shr    edx, 1
                             <1> ;     mov    ecx, 32/2
                             <1> ;ac97_set_bdl_buffer:
                             <1> ;     ; 1st half of DMA buffer
                             <1> ;     mov    eax, [audio_dma_buff]
                             <1> ;     push   eax
                             <1> ;     stosd
                             <1> ;     mov    eax, edx ; dma buffer size / 2
                             <1> ;     or     eax, IOC+BUP
                             <1> ;     stosd
                             <1> ;     pop    eax
                             <1> ;     ; 2nd half of DMA buffer
                             <1> ;     add    eax, edx
                             <1> ;     stosd
                             <1> ;     mov    eax, edx ; dma buffer size / 2
                             <1> ;     or     eax, IOC+BUP
                             <1> ;     stosd
                             <1> ;     loop   ac97_set_bdl_buffer
                             <1>
                             <1> ; tell the DMA engine where to find our list of Buffer Descriptors.
                             <1> ; this 32bit value is a flat mode memory offset (ie no segment:offset)
                             <1> ;
                             <1> ; write NABMBAR+10h with offset of buffer descriptor list
                             <1> ;
000123F1 B8[F86B0100]        <1>         mov  eax, audio_bdl_buff
000123F6 668B15[F66B0100]    <1>       mov    dx, [NABMBAR]
000123FD 6683C210            <1>       add    dx, PO_BDBAR_REG
00012401 EF                  <1>       out    dx, eax
                             <1> ;
                             <1> ; All set.  Let's play some music.
                             <1> ;
                             <1> ;
00012402 B81F000000          <1>       mov    eax, 31
00012407 E816000000          <1>       call    set_ac97_LastValidIndex
                             <1>
0001240C C605[F06B0100]01    <1>       mov    byte [audio_play_cmd], 1 ; play command (do not stop) !
                             <1>
                             <1> ac97_play: ; continue to play (after pause)
                             <1>       ; 11/06/2017
                             <1>       ; 29/05/2017
                             <1>       ; 28/05/2017
00012413 668B15[F66B0100]    <1>         mov     dx, [NABMBAR]
0001241A 6683C21B            <1>         add     dx, PO_CR_REG          ; PCM out control register
0001241E B011                <1>         mov al, IOCE+RPBM ; 29/05/2017
                             <1>          ;mov al, 1Dh ; (Ref: KolibriOS, intelac97.asm, 'play:')
00012420 EE                  <1>       out     dx, al                   ; set start!
                             <1>
                             <1>       ;mov   byte [audio_play_cmd], 1 ; play command (do not stop) !
                             <1>
00012421 C3                  <1>       retn
                             <1>
                             <1> ;input AL = index # to stop on
                             <1> set_ac97_LastValidIndex:
                             <1>       ; 28/05/2017
                             <1>       ; Derived from 'setLastValidIndex' procedure in 'ICHWAV.ASM'
                             <1>       ; .wav player for DOS by Jeff Leyda (02/09/2002)
00012422 668B15[F66B0100]    <1>       mov    dx, [NABMBAR]
00012429 6683C215            <1>       add    dx, PO_LVI_REG
0001242D EE                  <1>         out     dx, al
                             <1>       ;mov   [audio_lvi], al ; for ac97_int_handler
0001242E C3                  <1>       retn
                             <1>
                             <1> ac97_volume:
                             <1>       ; 28/05/2017
                             <1>       ; bl = component (0 = master/playback/lineout volume)
                             <1>       ; cl = left channel volume level (0 to 31)
                             <1>       ; ch = right channel volume level (0 to 31)
                             <1>
0001242F 08DB                <1>       or     bl, bl
00012431 7523                <1>       jnz    short ac97_vol_1 ; temporary !
00012433 66B81F1F            <1>       mov    ax, 1F1Fh ; 31,31
00012437 38C1                <1>       cmp    cl, al
00012439 771B                <1>       ja     short ac97_vol_1 ; temporary !
0001243B 38E5                <1>       cmp    ch, ah
0001243D 7717                <1>       ja     short ac97_vol_1 ; temporary !
0001243F 66890D[F26B0100]    <1>       mov    [audio_master_volume], cx
00012446 6629C8              <1>       sub    ax, cx
00012449 668B15[F46B0100]    <1>       mov      dx, [NAMBAR]
00012450 6683C202            <1>       add      dx, CODEC_MASTER_VOL_REG  ; 02h ; Line Out
00012454 66EF                <1>       out      dx, ax
                             <1> ac97_vol_1:
```

```
00012456 C3                  <1>       retn
                             <1> 
                             <1> ac97_int_handler:
                             <1>       ; 12/10/2017
                             <1>       ; 10/10/2017
                             <1>       ; 09/10/2017
                             <1>       ; 13/06/2017, 13/06/2017
                             <1>       ; 10/06/2017, 11/06/2017
                             <1>       ; Interrupt Handler for AC97 (ICH) Audio Controller
                             <1>       ; Note: called by 'dev_IRQ_service'
                             <1>       ; 28/05/2017
                             <1> 
                             <1>       ;push  eax ; * must be saved !
                             <1>       ;push  edx
                             <1>       ;push  ecx
                             <1>       ;push  ebx ; * must be saved !
                             <1>       ;push  esi
                             <1>       ;push  edi
                             <1> 
                             <1>       ;cmp   byte [audio_busy], 1
                             <1>       ;jnb   _ac97_ih2    ; busy !
                             <1> 
00012457 66BA3000            <1>         mov     dx, GLOB_STS_REG
0001245B 660315[F66B0100]    <1>         add dx, [NABMBAR]
00012462 ED                  <1>       in     eax, dx
                             <1> 
00012463 83F8FF              <1>         cmp     eax, 0FFFFFFFFh ; -1
00012466 0F849A000000        <1>         je      _ac97_ih3 ; exit
                             <1> 
0001246C A940000000          <1>         test    eax, 40h ; PCM Out Interrupt
00012471 750E                <1>         jnz     short _ac97_ih0
                             <1> 
00012473 85C0                <1>       test   eax, eax
00012475 0F848B000000        <1>       jz     _ac97_ih3 ; exit
                             <1> 
                             <1>       ;mov   dx, GLOB_STS_REG
                             <1>         ;add dx, [NABMBAR]
0001247B EF                  <1>       out    dx, eax
                             <1> 
0001247C E985000000          <1>       jmp    _ac97_ih3 ; exit
                             <1> 
                             <1> _ac97_ih0:
00012481 50                  <1>       push   eax
                             <1>       ; 09/10/2017
00012482 803D[F06B0100]01    <1>       cmp    byte [audio_play_cmd], 1
00012489 727C                <1>       jb     short _ac97_ih4 ; stop command !
                             <1> 
                             <1>       ;mov   byte [audio_busy], 1
                             <1> 
                             <1>       ;mov   al, 10h
                             <1>       ;mov   dx, PO_CR_REG
                             <1>         ;add dx, [NABMBAR]
                             <1>       ;out   dx, al
                             <1> 
0001248B 66B81C00            <1>       mov    ax, 1Ch ; FIFOE(=16)+BCIS(=8)+LVBCI(=4)
0001248F 66BA1600            <1>       mov    dx, PO_SR_REG
00012493 660315[F66B0100]    <1>         add dx, [NABMBAR]
0001249A 66EF                <1>       out    dx, ax
                             <1> 
0001249C 66BA1400            <1>       mov    dx, PO_CIV_REG
000124A0 660315[F66B0100]    <1>         add dx, [NABMBAR]
000124A7 EC                  <1>       in     al, dx
                             <1> 
                             <1>       ;cmp   al, [audio_civ] ; [audio_flag]
                             <1>       ;je    short _ac97_ih2
                             <1> 
000124A8 A2[F16B0100]        <1>       mov    [audio_civ], al
000124AD FEC8                <1>       dec    al
                             <1>       ;inc   al ; 11/06/2017
000124AF 241F                <1>       and    al, 1Fh
                             <1> 
000124B1 66BA1500            <1>         mov     dx, PO_LVI_REG
000124B5 660315[F66B0100]    <1>         add dx, [NABMBAR]
000124BC EE                  <1>         out dx, al
                             <1> 
                             <1>       ; 12/10/2017
000124BD A0[F16B0100]        <1>       mov    al, [audio_civ]
000124C2 FEC0                <1>       inc    al
000124C4 2401                <1>       and    al, 1
000124C6 A2[E46B0100]        <1>       mov    [audio_flag], al
                             <1>       ;; [audio_flag] : 0 = Buffer 1, 1 = Buffer 2
                             <1>       ;
000124CB 58                  <1>       pop    eax
                             <1>       ;
000124CC 83E040              <1>       and     eax, 40h
000124CF 668B15[F66B0100]    <1>         mov dx, [NABMBAR]
000124D6 6683C230            <1>       add    dx, GLOB_STS_REG
000124DA EF                  <1>       out    dx, eax
                             <1> 
                             <1>       ;; 13/06/2017
                             <1>       ;mov   al, 11h ; IOCE + RPBM
                             <1>       ;mov   dx, PO_CR_REG
                             <1>         ;add dx, [NABMBAR]
                             <1>       ;out   dx, al
                             <1> 
                             <1> ac97_tuneloop:
                             <1>       ; 09/10/2017
000124DB 8B3D[DC6B0100]      <1>       mov    edi, [audio_dma_buff]
000124E1 8B0D[E06B0100]      <1>       mov    ecx, [audio_dmabuff_size]
000124E7 D1E9                <1>       shr    ecx, 1 ; dma buff size / 2 = half buffer size
                             <1> 
                             <1>       ; 12/10/2017
000124E9 803D[E46B0100]00    <1>       cmp    byte [audio_flag], 0
```

```
000124F0 7702                <1>       ja     short _ac97_ih1  ; Playing Half Buffer 2 (Current: FLAG)
                             <1>       ; Playing Half Buffer 1 (Current: EOL)
000124F2 01CF                <1>       add    edi, ecx
                             <1> _ac97_ih1:
                             <1>       ; Update half buffer 2 while playing half buffer 1 (next: FLAG)
                             <1>       ; Update half buffer 1 while playing half buffer 2 (next: EOL)
                             <1>
000124F4 8B35[D46B0100]      <1>       mov    esi, [audio_p_buffer] ; phy addr of audio buff
000124FA C1E902              <1>       shr    ecx, 2 ; half buff size / 4
000124FD F3A5                <1>       rep    movsd
                             <1>
                             <1>       ; 10/10/2017
                             <1>       ; switch flag value
000124FF 8035[E46B0100]01    <1>       xor    byte [audio_flag], 1
                             <1>       ; 12/10/2017
                             <1>       ; [audio_flag] = 0 : Playing dma half buffer 2 (even index value)
                             <1>                       ; Next buffer (to update) is dma half buff 1
                             <1>       ;             = 1 : Playing dma half buffer 1 (odd index value)
                             <1>                       ; Next buffer (to update) is dma half buff 2
                             <1>
                             <1> _ac97_ih2:
                             <1>       ;mov   byte [audio_busy], 0
                             <1> _ac97_ih3:
                             <1>       ;pop   edi
                             <1>       ;pop   esi
                             <1>       ;pop   ebx ; * must be restored !
                             <1>       ;pop   ecx
                             <1>       ;pop   edx
                             <1>       ;pop   eax ; * must be restored !
                             <1>
00012506 C3                  <1>       retn
                             <1>
                             <1> _ac97_ih4:
                             <1>       ; 09/10/2017
00012507 E818000000          <1>       call   _ac97_stop
                             <1>       ;
0001250C 58                  <1>       pop    eax
                             <1>       ;
0001250D 83E040              <1>       and     eax, 40h
00012510 668B15[F66B0100]    <1>         mov dx, [NABMBAR]
00012517 6683C230            <1>       add    dx, GLOB_STS_REG
0001251B EF                  <1>       out    dx, eax
                             <1>
                             <1>       ;; 13/06/2017
                             <1>       ;mov   al, 11h ; IOCE + RPBM
                             <1>       ;dx, PO_CR_REG
                             <1>         ;add dx, [NABMBAR]
                             <1>       ;out   dx, al
                             <1>
                             <1>       ; 10/10/2017
                             <1>       ;jmp   short _ac97_ih3  ; exit
0001251C C3                  <1>       retn
                             <1>
                             <1> ac97_stop:
                             <1>       ; 28/05/2017
0001251D C605[F06B0100]00    <1>       mov    byte [audio_play_cmd], 0 ; stop !
                             <1> _ac97_stop: ; 09/10/2017
                             <1>       ; 29/05/2017
                             <1>       ;mov   dx, [NABMBAR]
                             <1>       ;add   dx, PO_CR_REG
                             <1>       ;mov   al, 0
                             <1>       ;out   dx, al
                             <1>
                             <1>       ; 11/06/2017
00012524 30C0                <1>       xor    al, al ; 0
00012526 E813000000          <1>       call   ac97_po_cmd
                             <1>
                             <1>       ; (Ref: KolibriOS, intelac97.asm, 'stop:')
                             <1>       ; Clear FIFOE, BCIS, LVBCI (Ref: Intel ICH hub manual)
0001252B 66B81C00            <1>       mov     ax, 1Ch
0001252F 668B15[F66B0100]    <1>       mov     dx, [NABMBAR]
00012536 6683C216            <1>       add     dx, PO_SR_REG
0001253A 66EF                <1>       out     dx, ax
                             <1>
                             <1>       ;retn
                             <1>
                             <1>       ; 11/06/2017
0001253C B002                <1>       mov     al, RR
                             <1> ac97_po_cmd:
                             <1>        ;11/06/2017
                             <1>       ; 29/05/2017
0001253E 668B15[F66B0100]    <1>       mov     dx, [NABMBAR]
00012545 6683C21B            <1>         add     dx, PO_CR_REG          ; PCM out control register
00012549 EE                  <1>       out    dx, al
0001254A C3                  <1>       retn
                             <1>
                             <1> ac97_pause:
                             <1>       ; 11/06/2017
                             <1>       ; 29/05/2017
0001254B B010                <1>       mov    al, IOCE
0001254D EBEF                <1>       jmp    short ac97_po_cmd
                             <1>
                             <1> reset_ac97_controller:
                             <1>       ; 10/06/2017
                             <1>       ; 29/05/2017
                             <1>       ; 28/05/2017
                             <1>       ; reset AC97 audio controller registers
0001254F 31C0                <1>       xor     eax, eax
00012551 66BA0B00            <1>         mov dx, PI_CR_REG
00012555 660315[F66B0100]    <1>       add    dx, [NABMBAR]
0001255C EE                  <1>       out     dx, al
                             <1>
0001255D 66BA1B00            <1>         mov     dx, PO_CR_REG
```

```
00012561 660315[F66B0100]    <1>       add    dx, [NABMBAR]
00012568 EE                  <1>       out     dx, al
                             <1> 
00012569 66BA2B00            <1>         mov     dx, MC_CR_REG
0001256D 660315[F66B0100]    <1>       add    dx, [NABMBAR]
00012574 EE                  <1>       out     dx, al
                             <1> 
00012575 B002                <1>         mov     al, RR
00012577 66BA0B00            <1>         mov     dx, PI_CR_REG
0001257B 660315[F66B0100]    <1>       add    dx, [NABMBAR]
00012582 EE                  <1>       out     dx, al
                             <1> 
00012583 66BA1B00            <1>         mov     dx, PO_CR_REG
00012587 660315[F66B0100]    <1>       add    dx, [NABMBAR]
0001258E EE                  <1>       out     dx, al
                             <1> 
0001258F 66BA2B00            <1>         mov     dx, MC_CR_REG
00012593 660315[F66B0100]    <1>       add    dx, [NABMBAR]
0001259A EE                  <1>       out     dx, al
                             <1> 
0001259B C3                  <1>       retn
                             <1> 
                             <1> ac97_reset:
                             <1>       ; 10/06/2017
                             <1>       ; 29/05/2017
                             <1>       ; 28/05/2017
0001259C E8AEFFFFFF          <1>       call   reset_ac97_controller
                             <1>       ; 29/05/2017
                             <1>       ;jmp   reset_ac97_codec
                             <1> reset_ac97_codec:
                             <1>       ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, intelac97.asm)
000125A1 66BA2C00            <1>       mov    dx, GLOB_CNT_REG ; 2Ch
000125A5 660315[F66B0100]    <1>       add    dx, [NABMBAR]
000125AC ED                  <1>       in     eax, dx
                             <1> 
000125AD A902000000          <1>       test   eax, 2
000125B2 7407                <1>       jz     short _r_ac97codec_cold
                             <1> 
000125B4 E80F000000          <1>       call   warm_ac97codec_reset
000125B9 7308                <1>       jnc    short _r_ac97codec_ok
                             <1> _r_ac97codec_cold:
000125BB E83D000000          <1>         call    cold_ac97codec_reset
000125C0 7301                <1>         jnc     short _r_ac97codec_ok
                             <1> 
                             <1>       ; 16/04/2017
                             <1>         ;xor     eax, eax         ; timeout error
                             <1>              ;stc
000125C2 C3                  <1>       retn
                             <1> 
                             <1> _r_ac97codec_ok:
000125C3 31C0                <1>         xor     eax, eax
                             <1>         ;mov al, VIA_ACLINK_C00_READY ; 1
000125C5 FEC0                <1>         inc  al
000125C7 C3                  <1>       retn
                             <1> 
                             <1> warm_ac97codec_reset:
                             <1>       ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, intelac97.asm)
000125C8 B806000000          <1>         mov     eax, 6
000125CD 66BA2C00            <1>       mov    dx, GLOB_CNT_REG ; 2Ch
000125D1 660315[F66B0100]    <1>       add    dx, [NABMBAR]
000125D8 EF                  <1>       out    dx, eax
                             <1> 
000125D9 B90A000000          <1>       mov    ecx, 10      ; total 1s
                             <1> _warm_ac97c_rst_wait:
000125DE 51                  <1>       push   ecx
000125DF E8D6F5FFFF          <1>       call   delay_100ms
000125E4 59                  <1>       pop    ecx
                             <1> 
000125E5 66BA3000            <1>       mov    dx, GLOB_STS_REG ; 30h
000125E9 660315[F66B0100]    <1>       add    dx, [NABMBAR]
000125F0 ED                  <1>       in     eax, dx
                             <1> 
000125F1 A900030010          <1>       test   eax, CTRL_ST_CREADY
000125F6 7504                <1>       jnz    short _warm_ac97c_rst_ok
                             <1> 
000125F8 49                  <1>         dec     ecx
000125F9 75E3                <1>         jnz     short _warm_ac97c_rst_wait
                             <1> 
                             <1> _warm_ac97c_rst_fail:
000125FB F9                  <1>         stc
                             <1> _warm_ac97c_rst_ok:
000125FC C3                  <1>       retn
                             <1> 
                             <1> cold_ac97codec_reset:
                             <1>       ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, intelac97.asm)
000125FD B802000000          <1>         mov     eax, 2
00012602 66BA2C00            <1>       mov    dx, GLOB_CNT_REG ; 2Ch
00012606 660315[F66B0100]    <1>       add    dx, [NABMBAR]
0001260D EF                  <1>       out    dx, eax
                             <1> 
0001260E E8A7F5FFFF          <1>       call   delay_100ms  ; wait 100 ms
00012613 E8A2F5FFFF          <1>       call   delay_100ms  ; wait 100 ms
00012618 E89DF5FFFF          <1>       call   delay_100ms  ; wait 100 ms
0001261D E898F5FFFF          <1>       call   delay_100ms  ; wait 100 ms
                             <1> 
00012622 B910000000          <1>       mov    ecx, 16      ; total 20*100 ms = 2s
                             <1> _cold_ac97c_rst_wait:
00012627 66BA3000            <1>       mov    dx, GLOB_STS_REG ; 30h
0001262B 660315[F66B0100]    <1>       add    dx, [NABMBAR]
00012632 ED                  <1>       in     eax, dx
                             <1> 
00012633 A900030010          <1>       test   eax, CTRL_ST_CREADY
00012638 750B                <1>       jnz    short _cold_ac97c_rst_ok
```

```
                             <1>
0001263A 51                  <1>       push   ecx
0001263B E87AF5FFFF          <1>       call   delay_100ms
00012640 59                  <1>       pop    ecx
                             <1>
00012641 49                  <1>         dec     ecx
00012642 75E3                <1>         jnz     short _cold_ac97c_rst_wait
                             <1>
                             <1> _cold_ac97c_rst_fail:
00012644 F9                  <1>         stc
                             <1> _cold_ac97c_rst_ok:
00012645 C3                  <1>       retn
                             <1>
                             <1> sb16_current_sound_data:
                             <1>       ; 20/08/2017
                             <1>       ; 24/06/2017
                             <1>       ; 22/06/2017
                             <1>       ; get current sound (PCM out) data for graphics
                             <1>       ; (for Sound Blaster 16)
                             <1>       ; ebx = Physical address (on page boundary)
                             <1>       ; ecx = Byte count
                             <1>       ; [audio_buff_size]
                             <1>
                             <1>       ;;mov  edi, [audio_buff_size]
                             <1>       ;mov   edi, [audio_dmabuff_size]
                             <1>       ;mov   esi, [audio_dma_buff]
00012646 39CF                <1>       cmp    edi, ecx
00012648 7302                <1>       jnb    short sb16_gcd_0
0001264A 89F9                <1>       mov    ecx, edi
                             <1> sb16_gcd_0:
                             <1>       ; 20/08/2017
0001264C 803D[EC6B0100]10    <1>       cmp    byte [audio_bps], 16
00012653 750F                <1>       jne    short sb16_gcd_1 ; 8 bit DMA channel
00012655 E4C6                <1>       in     al, 0C6h ; DMA channel 5 count register
00012657 88C2                <1>       mov    dl, al
00012659 E4C6                <1>       in     al, 0C6h
0001265B 88C6                <1>       mov      dh, al
0001265D 0FB7C2              <1>       movzx  eax, dx
00012660 D1E0                <1>       shl    eax, 1 ; word count -> byte count
00012662 EB4E                <1>       jmp    short sb16_gcd_2
                             <1> sb16_gcd_1:
00012664 E403                <1>       in     al, 03h ; DMA channel 1 count register
00012666 88C2                <1>       mov    dl, al
00012668 E403                <1>       in     al, 03h
0001266A 88C6                <1>       mov      dh, al
0001266C 0FB7C2              <1>       movzx  eax, dx
0001266F EB41                <1>       jmp    short sb16_gcd_2
                             <1> ;sb16_gcd_2:
                             <1> ;     cmp    eax, ecx
                             <1> ;     jnb    short sb16_gcd_3
                             <1> ;     ; remain count < graphics bytes
                             <1> ;     mov    eax, ecx ; fix remain count to data size
                             <1> ;sb16_gcd_3:
                             <1> ;     sub    edi, eax
                             <1> ;     jna    short sb16_gcd_4
                             <1> ;     add    esi, edi ; dma buffer offset
                             <1> ;sb16_gcd_4:
                             <1> ;     mov    edi, ebx ; buffer address (for graphics)
                             <1> ;     mov    [u.r0], ecx
                             <1> ;     rep    movsb
                             <1> ;     retn
                             <1>
                             <1> get_current_sound_data:
                             <1>       ; 24/06/2017
                             <1>       ; 22/06/2017
                             <1>       ; get current sound (PCM out) data for graphics
                             <1>       ;
                             <1>       ; ebx = Physical address (on page boundary)
                             <1>       ; ecx = Byte count
                             <1>       ; [audio_buff_size]
                             <1>
                             <1>       ;mov   edi, [audio_buff_size]
00012671 8B3D[E06B0100]      <1>       mov    edi, [audio_dmabuff_size]
00012677 8B35[DC6B0100]      <1>       mov    esi, [audio_dma_buff]
0001267D 803D[BD6B0100]02    <1>       cmp    byte [audio_device], 2
00012684 72C0                <1>       jb     short sb16_current_sound_data ; = 1
00012686 D1EF                <1>       shr    edi, 1
00012688 39CF                <1>       cmp    edi, ecx
0001268A 7302                <1>       jnb    short gcd_0
0001268C 89F9                <1>       mov    ecx, edi
                             <1> gcd_0:
0001268E 803D[BD6B0100]03    <1>       cmp    byte [audio_device], 3
00012695 7232                <1>       jb     short ac97_current_sound_data ; = 2
                             <1>       ; = 3
                             <1> vt8233_current_sound_data:
                             <1>       ; 22/06/2017
                             <1>       ; 21/06/2017
                             <1>       ; get current sound (PCM out) data for graphics
                             <1>       ; (for VT 8233, VT 8237R)
                             <1>       ; ebx = Physical address (on page boundary)
                             <1>       ; ecx = Byte count
                             <1>       ; [audio_buff_size]
                             <1>
                             <1>       ;;mov  edi, [audio_buff_size]
                             <1>       ;mov   edi, [audio_dmabuff_size]
                             <1>       ;mov   esi, [audio_dma_buff]
                             <1>       ;shr   edi, 1
                             <1>       ;cmp   edi, ecx
                             <1>       ;jnb   short vt8233_gcd_1
                             <1>       ;mov   ecx, edi
                             <1> vt8233_gcd_1:
00012697 BA0C000000          <1>       mov    edx, VIA_REG_OFFSET_CURR_COUNT
0001269C E879F5FFFF          <1>       call   ctrl_io_r32
```

```
000126A1 89C2                     <1>       mov    edx, eax ; remain count (bits 23-0),
                                  <1>                      ; SGD index (bits 31-24)
000126A3 81E200000001             <1>       and    edx, 1000000h ; SGD index (0 = 1st half)
000126A9 7402                     <1>       jz     short vt8233_gcd_2
                                  <1>       ; the second half of DMA buffer
000126AB 01FE                     <1>       add    esi, edi
                                  <1> vt8233_gcd_2:
000126AD 25FFFFFF00               <1>       and    eax, 0FFFFFFh ; bits 23-0
                                  <1> ac97_gcd_2:
                                  <1> sb16_gcd_2:
000126B2 39C8                     <1>       cmp    eax, ecx
000126B4 7302                     <1>       jnb    short vt8233_gcd_3
                                  <1>       ; remain count < graphics bytes
000126B6 89C8                     <1>       mov    eax, ecx ; fix remain count to data size
                                  <1> vt8233_gcd_3:
000126B8 29C7                     <1>       sub    edi, eax
000126BA 7602                     <1>       jna    short vt8233_gcd_4
000126BC 01FE                     <1>       add    esi, edi ; dma buffer offset
                                  <1> vt8233_gcd_4:
000126BE 89DF                     <1>       mov    edi, ebx ; buffer address (for graphics)
000126C0 890D[64030300]           <1>       mov    [u.r0], ecx
000126C6 F3A4                     <1>       rep    movsb
                                  <1> vt8233_gcd_5:
000126C8 C3                       <1>       retn
                                  <1>
                                  <1> ac97_current_sound_data:
                                  <1>       ; 23/06/2017
                                  <1>       ; 22/06/2017
                                  <1>       ; get current sound (PCM out) data for graphics
                                  <1>       ; (for AC'97, ICH)
                                  <1>       ; ebx = Physical address (on page boundary)
                                  <1>       ; ecx = Byte count
                                  <1>       ; [audio_buff_size]
                                  <1>
                                  <1>       ;;mov  edi, [audio_buff_size]
                                  <1>       ;mov   edi, [audio_dmabuff_size]
                                  <1>       ;mov   esi, [audio_dma_buff]
                                  <1>       ;shr   edi, 1
                                  <1>       ;cmp   edi, ecx
                                  <1>       ;jnb   short ac97_gcd_0
                                  <1>       ;mov   ecx, edi
                                  <1> ac97_gcd_0:
000126C9 66BA1400                 <1>       mov    dx, PO_CIV_REG ; Position In Current Buff Reg
000126CD 660315[F66B0100]         <1>       add    dx, [NABMBAR]
000126D4 EC                       <1>       in     al, dx ; current index value
000126D5 A801                     <1>       test   al, 1
000126D7 7402                     <1>       jz     short ac97_gcd_1
000126D9 01FE                     <1>       add    esi, edi
                                  <1> ac97_gcd_1:
000126DB 31C0                     <1>       xor    eax, eax
000126DD 66BA1800                 <1>       mov    dx, PO_PICB_REG ; Position In Current Buff Reg
000126E1 660315[F66B0100]         <1>       add    dx, [NABMBAR]
000126E8 66ED                     <1>       in     ax, dx ; remain dwords
000126EA C1E002                   <1>       shl    eax, 2 ; remain bytes ; 23/06/2017
000126ED EBC3                     <1>       jmp    short ac97_gcd_2
                                  <1> ;     cmp    eax, ecx
                                  <1> ;     jnb    short ac97_gcd_2
                                  <1> ;     ; remain count < graphics bytes
                                  <1> ;     mov    eax, ecx ; fix remain count to data size
                                  <1> ;ac97_gcd_2:
                                  <1> ;     sub    edi, eax
                                  <1> ;     jna    short ac97_gcd_3
                                  <1> ;     add    esi, edi ; dma buffer offset
                                  <1> ;ac97_gcd_3:
                                  <1> ;     mov    edi, ebx ; buffer address (for graphics)
                                  <1> ;     mov    [u.r0], ecx
                                  <1> ;     rep    movsb
                                  <1> ;     retn
                                  <1>
                                  <1> sb16_get_dma_buff_off:
                                  <1>       ; 28/10/2017
                                  <1>       ; 24/06/2017
                                  <1>       ; 22/06/2017
                                  <1>       ; get current (PCM OUT DMA buffer) pointer
                                  <1>       ; (for Sound Blaster 16)
                                  <1>
                                  <1>       ;mov   ecx, [audio_dmabuff_size]
                                  <1>       ;xor   ebx, ebx
                                  <1>       ;shr   ecx, 1
                                  <1> sb16_gdmabo_0:
                                  <1>       ; 28/10/2017
000126EF 803D[EC6B0100]10         <1>       cmp    byte [audio_bps], 16
000126F6 750F                     <1>       jne    short sb16_gdmabo_1 ; 8 bit DMA channel
                                  <1>       ; 16 bit DMA channel
000126F8 E4C6                     <1>       in     al, 0C6h ; DMA channel 5 count register
000126FA 88C2                     <1>       mov    dl, al
000126FC E4C6                     <1>       in     al, 0C6h
000126FE 88C6                     <1>       mov     dh, al
00012700 0FB7C2                   <1>       movzx  eax, dx
00012703 D1E0                     <1>       shl    eax, 1 ; word count -> byte count
00012705 EB3D                     <1>       jmp    short sb16_gdmabo_2
                                  <1> sb16_gdmabo_1:
00012707 E403                     <1>       in     al, 03h ; DMA channel 1 count register
00012709 88C2                     <1>       mov    dl, al
0001270B E403                     <1>       in     al, 03h
0001270D 88C6                     <1>       mov     dh, al
0001270F 0FB7C2                   <1>       movzx  eax, dx
00012712 EB30                     <1>       jmp    short sb16_gdmabo_2
                                  <1>
                                  <1> get_dma_buffer_offset:
                                  <1>       ; 24/06/2017
                                  <1>       ; 22/06/2017
                                  <1>       ; get current sound (PCM out) data for graphics
```

```
                              <1>       ;
                              <1>       ; ebx = Physical address (on page boundary)
                              <1>       ; ecx = Byte count
                              <1>       ; [audio_buff_size]
                              <1>
00012714 8B0D[E06B0100]       <1>       mov    ecx, [audio_dmabuff_size]
0001271A 31DB                 <1>       xor    ebx, ebx
                              <1> gdmabo_0:
0001271C 803D[BD6B0100]02     <1>       cmp    byte [audio_device], 2
00012723 72CA                 <1>       jb     short sb16_get_dma_buff_off
00012725 742A                 <1>       je     short ac97_get_dma_buff_off
                              <1>
                              <1> vt8233_get_dma_buff_off:
                              <1>       ; 24/06/2017
                              <1>       ; 22/06/2017
                              <1>       ; get current (PCM OUT DMA buffer) pointer
                              <1>       ; (for VT 8233, VT 8237R)
                              <1>
                              <1>       ;mov   ecx, [audio_dmabuff_size]
                              <1>       ;xor   ebx, ebx
00012727 D1E9                 <1>       shr    ecx, 1
                              <1> vt8233_gdmabo_0:
00012729 BA0C000000           <1>       mov    edx, VIA_REG_OFFSET_CURR_COUNT
0001272E E8E7F4FFFF           <1>       call   ctrl_io_r32
00012733 89C2                 <1>       mov    edx, eax ; remain count (bits 23-0),
                              <1>                      ; SGD index (bits 31-24)
00012735 81E200000001         <1>       and    edx, 1000000h ; SGD index (0 = 1st half)
0001273B 7402                 <1>       jz     short vt8233_gdmabo_1
                              <1>       ; the second half of DMA buffer
0001273D 89CB                 <1>       mov    ebx, ecx
                              <1>
                              <1> vt8233_gdmabo_1:
0001273F 25FFFFFF00           <1>       and    eax, 0FFFFFFh ; bits 23-0
                              <1> sb16_gdmabo_2:
                              <1> ac97_gdmabo_2:
00012744 29C1                 <1>       sub    ecx, eax
00012746 7602                 <1>       jna    short vt8233_gdmabo_2
00012748 01CB                 <1>       add    ebx, ecx ; dma buffer offset
                              <1> vt8233_gdmabo_2:
0001274A 891D[64030300]       <1>       mov    [u.r0], ebx
00012750 C3                   <1>       retn
                              <1>
                              <1> ac97_get_dma_buff_off:
                              <1>       ; 24/06/2017
                              <1>       ; 22/06/2017
                              <1>       ; get current (PCM OUT DMA buffer) pointer
                              <1>       ; (for AC'97, ICH)
                              <1>       ; ebx = Physical address (on page boundary)
                              <1>       ; ecx = Byte count
                              <1>       ; [audio_buff_size]
                              <1>
                              <1>       ;mov   ecx, [audio_dmabuff_size]
                              <1>       ;xor   ebx, ebx
00012751 D1E9                 <1>       shr    ecx, 1
                              <1> ac97_gdmabo_0:
00012753 66BA1400             <1>       mov    dx, PO_CIV_REG ; Position In Current Buff Reg
00012757 660315[F66B0100]     <1>       add    dx, [NABMBAR]
0001275E EC                   <1>       in     al, dx ; current index value
0001275F A801                 <1>       test   al, 1
00012761 7402                 <1>       jz     short ac97_gdmabo_1
00012763 89CB                 <1>       mov    ebx, ecx
                              <1> ac97_gdmabo_1:
00012765 31C0                 <1>       xor    eax, eax
00012767 66BA1800             <1>       mov    dx, PO_PICB_REG ; Position In Current Buff Reg
0001276B 660315[F66B0100]     <1>       add    dx, [NABMBAR]
00012772 66ED                 <1>       in     ax, dx ; remain dwords
00012774 EBCE                 <1>       jmp    short ac97_gdmabo_2

00012776 90<rept>                 align 4

                                  %include 'vgadata.s' ; 04/07/2016
                             <1> ; ***********************************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - vgadata.s  (palette and fond data)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 04/07/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 16/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Turkish Rational DOS
                             <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
                             <1> ;
                             <1> ; Derived from 'Plex86/Bochs VGABios' source code, vgabios-0.7a (2011)
                             <1> ; by the LGPL VGABios Developers Team (2001-2008), 'vgatables.h'
                             <1> ;
                             <1> ; Oracle VirtualBox 5.0.24 VGABios Source Code
                             <1> ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
                             <1> ;
                             <1> ; Palette and font data in assembly language format:
                             <1> ; 'VBoxVgaBiosAlternative.asm'
                             <1>
                             <1> ; ***********************************************************************************************
                             <1>
                             <1> ; 04/07/2016
                             <1> ; COLOR DATA
                             <1>
                             <1> palette0:
00012778 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
00012781 00000000000000      <1>
00012788 00000000000000002A- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
00012791 2A2A2A2A2A2A2A      <1>
00012798 2A2A2A2A2A2A2A2A2A- <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
000127A1 2A2A2A2A2A2A2A      <1>
000127A8 2A2A2A2A2A2A2A2A2A- <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
000127B1 2A2A2A2A2A2A2A      <1>
```

```
32 000127B8 2A2A2A2A2A2A2A2A3F- <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
32 000127C1 3F3F3F3F3F3F3F      <1>
33 000127C8 3F3F3F3F3F3F3F3F3F- <1>     db  03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
33 000127D1 3F3F3F3F3F3F3F      <1>
34 000127D8 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
34 000127E1 00000000000000      <1>
35 000127E8 00000000000000002A- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
35 000127F1 2A2A2A2A2A2A2A      <1>
36 000127F8 2A2A2A2A2A2A2A2A2A- <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
36 00012801 2A2A2A2A2A2A2A      <1>
37 00012808 2A2A2A2A2A2A2A2A2A- <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
37 00012811 2A2A2A2A2A2A2A      <1>
38 00012818 2A2A2A2A2A2A2A2A3F- <1>     db  02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
38 00012821 3F3F3F3F3F3F3F      <1>
39 00012828 3F3F3F3F3F3F3F3F3F- <1>     db  03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
39 00012831 3F3F3F3F3F3F3F      <1>
40                              <1> palette1:
41 00012838 00000000002A002A00- <1>     db  000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
41 00012841 002A2A2A00002A      <1>
42 00012848 002A2A15002A2A2A00- <1>     db  000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah
42 00012851 000000002A002A      <1>
43 00012858 00002A2A2A00002A00- <1>     db  000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah, 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah
43 00012861 2A2A15002A2A2A      <1>
44 00012868 15151515153F153F15- <1>     db  015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh, 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh
44 00012871 153F3F3F15153F      <1>
45 00012878 153F3F3F153F3F3F15- <1>     db  015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
45 00012881 151515153F153F      <1>
46 00012888 15153F3F3F15153F15- <1>     db  015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
46 00012891 3F3F3F153F3F3F      <1>
47 00012898 00000000002A002A00- <1>     db  000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
47 000128A1 002A2A2A00002A      <1>
48 000128A8 002A2A15002A2A2A00- <1>     db  000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah
48 000128B1 000000002A002A      <1>
49 000128B8 00002A2A2A00002A00- <1>     db  000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah, 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah
49 000128C1 2A2A15002A2A2A      <1>
50 000128C8 15151515153F153F15- <1>     db  015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh, 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh
50 000128D1 153F3F3F15153F      <1>
51 000128D8 153F3F3F153F3F3F15- <1>     db  015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
51 000128E1 151515153F153F      <1>
52 000128E8 15153F3F3F15153F15- <1>     db  015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
52 000128F1 3F3F3F153F3F3F      <1>
53                              <1> palette2:
54 000128F8 00000000002A002A00- <1>     db  000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
54 00012901 002A2A2A00002A      <1>
55 00012908 002A2A2A002A2A2A00- <1>     db  000h, 02ah, 02ah, 02ah, 000h, 02ah, 02ah, 02ah, 000h, 000h, 015h, 000h, 000h, 03fh, 000h, 02ah
55 00012911 001500003F002A      <1>
56 00012918 15002A3F2A00152A00- <1>     db  015h, 000h, 02ah, 03fh, 02ah, 000h, 015h, 02ah, 000h, 03fh, 02ah, 02ah, 015h, 02ah, 02ah, 03fh
56 00012921 3F2A2A152A2A3F      <1>
57 00012928 00150000152A003F00- <1>     db  000h, 015h, 000h, 000h, 015h, 02ah, 000h, 03fh, 000h, 000h, 03fh, 02ah, 02ah, 015h, 000h, 02ah
57 00012931 003F2A2A15002A      <1>
58 00012938 152A2A3F002A3F2A00- <1>     db  015h, 02ah, 02ah, 03fh, 000h, 02ah, 03fh, 02ah, 000h, 015h, 015h, 000h, 015h, 03fh, 000h, 03fh
58 00012941 151500153F003F      <1>
59 00012948 15003F3F2A15152A15- <1>     db  015h, 000h, 03fh, 03fh, 02ah, 015h, 015h, 02ah, 015h, 03fh, 02ah, 03fh, 015h, 02ah, 03fh, 03fh
59 00012951 3F2A3F152A3F3F      <1>
60 00012958 15000015002A152A00- <1>     db  015h, 000h, 000h, 015h, 000h, 02ah, 015h, 02ah, 000h, 015h, 02ah, 02ah, 03fh, 000h, 000h, 03fh
60 00012961 152A2A3F00003F      <1>
61 00012968 002A3F2A003F2A2A15- <1>     db  000h, 02ah, 03fh, 02ah, 000h, 03fh, 02ah, 02ah, 015h, 000h, 015h, 015h, 000h, 03fh, 015h, 02ah
61 00012971 001515003F152A      <1>
62 00012978 15152A3F3F00153F00- <1>     db  015h, 015h, 02ah, 03fh, 03fh, 000h, 015h, 03fh, 000h, 03fh, 03fh, 02ah, 015h, 03fh, 02ah, 03fh
62 00012981 3F3F2A153F2A3F      <1>
63 00012988 15150015152A153F00- <1>     db  015h, 015h, 000h, 015h, 015h, 02ah, 015h, 03fh, 000h, 015h, 03fh, 02ah, 03fh, 015h, 000h, 03fh
63 00012991 153F2A3F15003F      <1>
64 00012998 152A3F3F003F3F2A15- <1>     db  015h, 02ah, 03fh, 03fh, 000h, 03fh, 03fh, 02ah, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
64 000129A1 151515153F153F      <1>
65 000129A8 15153F3F3F15153F15- <1>     db  015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
65 000129B1 3F3F3F153F3F3F      <1>
66                              <1> palette3:
67 000129B8 00000000002A002A00- <1>     db  000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
67 000129C1 002A2A2A00002A      <1>
68 000129C8 002A2A15002A2A2A15- <1>     db  000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
68 000129D1 151515153F153F      <1>
69 000129D8 15153F3F3F15153F15- <1>     db  015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
69 000129E1 3F3F3F153F3F3F      <1>
70 000129E8 000000050505080808- <1>     db  000h, 000h, 000h, 005h, 005h, 005h, 008h, 008h, 008h, 00bh, 00bh, 00bh, 00eh, 00eh, 00eh, 011h
70 000129F1 0B0B0B0E0E0E11      <1>
71 000129F8 11111414141818181C- <1>     db  011h, 011h, 014h, 014h, 014h, 018h, 018h, 018h, 01ch, 01ch, 01ch, 020h, 020h, 020h, 024h, 024h
71 00012A01 1C1C2020202424      <1>
72 00012A08 242828282D2D2D3232- <1>     db  024h, 028h, 028h, 028h, 02dh, 02dh, 02dh, 032h, 032h, 032h, 038h, 038h, 038h, 03fh, 03fh, 03fh
72 00012A11 323838383F3F3F      <1>
73 00012A18 00003F10003F1F003F- <1>     db  000h, 000h, 03fh, 010h, 000h, 03fh, 01fh, 000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h, 03fh, 03fh
73 00012A21 2F003F3F003F3F      <1>
74 00012A28 002F3F001F3F00103F- <1>     db  000h, 02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 000h, 000h, 03fh, 010h, 000h, 03fh, 01fh
74 00012A31 00003F10003F1F      <1>
75 00012A38 003F2F003F3F002F3F- <1>     db  000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h, 02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 000h
75 00012A41 001F3F00103F00      <1>
76 00012A48 003F00003F10003F1F- <1>     db  000h, 03fh, 000h, 000h, 03fh, 010h, 000h, 03fh, 01fh, 000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h
76 00012A51 003F2F003F3F00      <1>
77 00012A58 2F3F001F3F00103F1F- <1>     db  02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh, 02fh, 01fh
77 00012A61 1F3F271F3F2F1F      <1>
78 00012A68 3F371F3F3F1F3F3F1F- <1>     db  03fh, 037h, 01fh, 03fh, 03fh, 01fh, 03fh, 03fh, 01fh, 037h, 03fh, 01fh, 02fh, 03fh, 01fh, 027h
78 00012A71 373F1F2F3F1F27      <1>
79 00012A78 3F1F1F3F271F3F2F1F- <1>     db  03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh, 02fh, 01fh, 03fh, 037h, 01fh, 03fh, 03fh, 01fh, 037h
79 00012A81 3F371F3F3F1F37      <1>
80 00012A88 3F1F2F3F1F273F1F1F- <1>     db  03fh, 01fh, 02fh, 03fh, 01fh, 027h, 03fh, 01fh, 01fh, 03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh
80 00012A91 3F1F1F3F271F3F      <1>
81 00012A98 2F1F3F371F3F3F1F37- <1>     db  02fh, 01fh, 03fh, 037h, 01fh, 03fh, 03fh, 01fh, 037h, 03fh, 01fh, 02fh, 03fh, 01fh, 027h, 03fh
81 00012AA1 3F1F2F3F1F273F      <1>
82 00012AA8 2D2D3F312D3F362D3F- <1>     db  02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h, 02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh, 03fh, 03fh
82 00012AB1 3A2D3F3F2D3F3F      <1>
83 00012AB8 2D3A3F2D363F2D313F- <1>     db  02dh, 03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h
83 00012AC1 2D2D3F312D3F36      <1>
84 00012AC8 2D3F3A2D3F3F2D3A3F- <1>     db  02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh, 03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 02dh
84 00012AD1 2D363F2D313F2D      <1>
85 00012AD8 2D3F2D2D3F312D3F36- <1>     db  02dh, 03fh, 02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h, 02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh
85 00012AE1 2D3F3A2D3F3F2D      <1>
86 00012AE8 3A3F2D363F2D313F00- <1>     db  03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 000h, 000h, 01ch, 007h, 000h, 01ch, 00eh, 000h
86 00012AF1 001C07001C0E00      <1>
87 00012AF8 1C15001C1C001C1C00- <1>     db  01ch, 015h, 000h, 01ch, 01ch, 000h, 01ch, 01ch, 000h, 015h, 01ch, 000h, 00eh, 01ch, 000h, 007h
87 00012B01 151C000E1C0007      <1>
```

```
00012B08 1C00001C07001C0E00- <1>     db  01ch, 000h, 000h, 01ch, 007h, 000h, 01ch, 00eh, 000h, 01ch, 015h, 000h, 01ch, 01ch, 000h, 015h
00012B11 1C15001C1C0015      <1>
00012B18 1C000E1C00071C0000- <1>     db  01ch, 000h, 00eh, 01ch, 000h, 007h, 01ch, 000h, 000h, 01ch, 000h, 000h, 01ch, 007h, 000h, 01ch
00012B21 1C00001C07001C      <1>
00012B28 0E001C15001C1C0015- <1>     db  00eh, 000h, 01ch, 015h, 000h, 01ch, 01ch, 000h, 015h, 01ch, 000h, 00eh, 01ch, 000h, 007h, 01ch
00012B31 1C000E1C00071C      <1>
00012B38 0E0E1C110E1C150E1C- <1>     db  00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h, 00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh, 01ch, 01ch
00012B41 180E1C1C0E1C1C      <1>
00012B48 0E181C0E151C0E111C- <1>     db  00eh, 018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h
00012B51 0E0E1C110E1C15      <1>
00012B58 0E1C180E1C1C0E181C- <1>     db  00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh, 018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 00eh
00012B61 0E151C0E111C0E      <1>
00012B68 0E1C0E0E1C110E1C15- <1>     db  00eh, 01ch, 00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h, 00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh
00012B71 0E1C180E1C1C0E      <1>
00012B78 181C0E151C0E111C14- <1>     db  018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 014h, 014h, 01ch, 016h, 014h, 01ch, 018h, 014h
00012B81 141C16141C1814      <1>
00012B88 1C1A141C1C141C1C14- <1>     db  01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ch, 01ch, 014h, 01ah, 01ch, 014h, 018h, 01ch, 014h, 016h
00012B91 1A1C14181C1416      <1>
00012B98 1C14141C16141C1814- <1>     db  01ch, 014h, 014h, 01ch, 016h, 014h, 01ch, 018h, 014h, 01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ah
00012BA1 1C1A141C1C141A      <1>
00012BA8 1C14181C14161C1414- <1>     db  01ch, 014h, 018h, 01ch, 014h, 016h, 01ch, 014h, 014h, 01ch, 014h, 014h, 01ch, 016h, 014h, 01ch
00012BB1 1C14141C16141C      <1>
00012BB8 18141C1A141C1C141A- <1>     db  018h, 014h, 01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ah, 01ch, 014h, 018h, 01ch, 014h, 016h, 01ch
00012BC1 1C14181C14161C      <1>
00012BC8 000010040010080010- <1>     db  000h, 000h, 010h, 004h, 000h, 010h, 008h, 000h, 010h, 00ch, 000h, 010h, 010h, 000h, 010h, 010h
00012BD1 0C001010001010      <1>
00012BD8 000C10000810000410- <1>     db  000h, 00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 000h, 000h, 010h, 004h, 000h, 010h, 008h
00012BE1 00001004001008      <1>
00012BE8 00100C001010000C10- <1>     db  000h, 010h, 00ch, 000h, 010h, 010h, 000h, 00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 000h
00012BF1 00081000041000      <1>
00012BF8 001000001004001008- <1>     db  000h, 010h, 000h, 000h, 010h, 004h, 000h, 010h, 008h, 000h, 010h, 00ch, 000h, 010h, 010h, 000h
00012C01 00100C00101000      <1>
00012C08 0C1000081000041008- <1>     db  00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 008h, 008h, 010h, 00ah, 008h, 010h, 00ch, 008h
00012C11 08100A08100C08      <1>
00012C18 100E08101008101008- <1>     db  010h, 00eh, 008h, 010h, 010h, 008h, 010h, 010h, 008h, 00eh, 010h, 008h, 00ch, 010h, 008h, 00ah
00012C21 0E10080C10080A      <1>
00012C28 100808100A08100C08- <1>     db  010h, 008h, 008h, 010h, 00ah, 008h, 010h, 00ch, 008h, 010h, 00eh, 008h, 010h, 010h, 008h, 00eh
00012C31 100E081010080E      <1>
00012C38 10080C10080A100808- <1>     db  010h, 008h, 00ch, 010h, 008h, 00ah, 010h, 008h, 008h, 010h, 008h, 008h, 010h, 00ah, 008h, 010h
00012C41 100808100A0810      <1>
00012C48 0C08100E081010080E- <1>     db  00ch, 008h, 010h, 00eh, 008h, 010h, 010h, 008h, 00eh, 010h, 008h, 00ch, 010h, 008h, 00ah, 010h
00012C51 10080C10080A10      <1>
00012C58 0B0B100C0B100D0B10- <1>     db  00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh, 00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh, 010h, 010h
00012C61 0F0B10100B1010      <1>
00012C68 0B0F100B0D100B0C10- <1>     db  00bh, 00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh
00012C71 0B0B100C0B100D      <1>
00012C78 0B100F0B10100B0F10- <1>     db  00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh, 00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 00bh
00012C81 0B0D100B0C100B      <1>
00012C88 0B100B0B100C0B100D- <1>     db  00bh, 010h, 00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh, 00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh
00012C91 0B100F0B10100B      <1>
00012C98 0F100B0D100B0C1000- <1>     db  00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
00012CA1 00000000000000      <1>
00012CA8 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
00012CB1 00000000000000      <1>
                             <1>
                             <1>
                             <1> ; 04/07/2016
                             <1> ; FONT DATA
                             <1>
                             <1> CRT_CHAR_GEN:
                             <1> vgafont8:
00012CB8 00000000000000007E- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 081h, 0a5h, 081h, 0bdh, 099h, 081h, 07eh
00012CC1 81A581BD99817E      <1>
00012CC8 7EFFDBFFC3E7FF7E6C- <1>     db  07eh, 0ffh, 0dbh, 0ffh, 0c3h, 0e7h, 0ffh, 07eh, 06ch, 0feh, 0feh, 0feh, 07ch, 038h, 010h, 000h
00012CD1 FEFEFE7C381000      <1>
00012CD8 10387CFE7C38100038- <1>     db  010h, 038h, 07ch, 0feh, 07ch, 038h, 010h, 000h, 038h, 07ch, 038h, 0feh, 0feh, 07ch, 038h, 07ch
00012CE1 7C38FEFE7C387C      <1>
00012CE8 1010387CFE7C387C00- <1>     db  010h, 010h, 038h, 07ch, 0feh, 07ch, 038h, 07ch, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h
00012CF1 00183C3C180000      <1>
00012CF8 FFFFE7C3C3E7FFFF00- <1>     db  0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h
00012D01 3C664242663C00      <1>
00012D08 FFC399BDBD99C3FF0F- <1>     db  0ffh, 0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 00fh, 007h, 00fh, 07dh, 0cch, 0cch, 0cch, 078h
00012D11 070F7DCCCCCC78      <1>
00012D18 3C6666663C187E183F- <1>     db  03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 018h, 03fh, 033h, 03fh, 030h, 030h, 070h, 0f0h, 0e0h
00012D21 333F303070F0E0      <1>
00012D28 7F637F636367E6C099- <1>     db  07fh, 063h, 07fh, 063h, 063h, 067h, 0e6h, 0c0h, 099h, 05ah, 03ch, 0e7h, 0e7h, 03ch, 05ah, 099h
00012D31 5A3CE7E73C5A99      <1>
00012D38 80E0F8FEF8E0800002- <1>     db  080h, 0e0h, 0f8h, 0feh, 0f8h, 0e0h, 080h, 000h, 002h, 00eh, 03eh, 0feh, 03eh, 00eh, 002h, 000h
00012D41 0E3EFE3E0E0200      <1>
00012D48 183C7E18187E3C1866- <1>     db  018h, 03ch, 07eh, 018h, 018h, 07eh, 03ch, 018h, 066h, 066h, 066h, 066h, 066h, 000h, 066h, 000h
00012D51 66666666006600      <1>
00012D58 7FDBDB7B1B1B1B003E- <1>     db  07fh, 0dbh, 0dbh, 07bh, 01bh, 01bh, 01bh, 000h, 03eh, 063h, 038h, 06ch, 06ch, 038h, 0cch, 078h
00012D61 63386C6C38CC78      <1>
00012D68 000000007E7E7E0018- <1>     db  000h, 000h, 000h, 000h, 07eh, 07eh, 07eh, 000h, 018h, 03ch, 07eh, 018h, 07eh, 03ch, 018h, 0ffh
00012D71 3C7E187E3C18FF      <1>
00012D78 183C7E181818180018- <1>     db  018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h
00012D81 1818187E3C1800      <1>
00012D88 00180CFE0C18000000- <1>     db  000h, 018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 030h, 060h, 0feh, 060h, 030h, 000h, 000h
00012D91 3060FE60300000      <1>
00012D98 0000C0C0C0FE000000- <1>     db  000h, 000h, 0c0h, 0c0h, 0c0h, 0feh, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h
00012DA1 2466FF66240000      <1>
00012DA8 00183C7EFFFF000000- <1>     db  000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 000h, 000h, 000h, 0ffh, 0ffh, 07eh, 03ch, 018h, 000h, 000h
00012DB1 FFFF7E3C180000      <1>
00012DB8 000000000000000030- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 030h, 078h, 078h, 030h, 030h, 000h, 030h, 000h
00012DC1 78783030003000      <1>
00012DC8 6C6C6C00000000006C- <1>     db  06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch, 0feh, 06ch, 06ch, 000h
00012DD1 6CFE6CFE6C6C00      <1>
00012DD8 307CC0780CF8300000- <1>     db  030h, 07ch, 0c0h, 078h, 00ch, 0f8h, 030h, 000h, 000h, 0c6h, 0cch, 018h, 030h, 066h, 0c6h, 000h
00012DE1 C6CC183066C600      <1>
00012DE8 386C3876DCCC760060- <1>     db  038h, 06ch, 038h, 076h, 0dch, 0cch, 076h, 000h, 060h, 060h, 0c0h, 000h, 000h, 000h, 000h, 000h
00012DF1 60C00000000000      <1>
00012DF8 183060606030180060- <1>     db  018h, 030h, 060h, 060h, 060h, 030h, 018h, 000h, 060h, 030h, 018h, 018h, 018h, 030h, 060h, 000h
00012E01 30181818306000      <1>
00012E08 00663CFF3C66000000- <1>     db  000h, 066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 030h, 030h, 0fch, 030h, 030h, 000h, 000h
00012E11 3030FC30300000      <1>
00012E18 000000000030306000- <1>     db  000h, 000h, 000h, 000h, 000h, 030h, 030h, 060h, 000h, 000h, 000h, 0fch, 000h, 000h, 000h, 000h
00012E21 0000FC00000000      <1>
00012E28 000000000030300006- <1>     db  000h, 000h, 000h, 000h, 000h, 030h, 030h, 000h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h
00012E31 0C183060C08000      <1>
```

```
00012E38 7CC6CEDEF6E67C0030- <1>      db  07ch, 0c6h, 0ceh, 0deh, 0f6h, 0e6h, 07ch, 000h, 030h, 070h, 030h, 030h, 030h, 030h, 0fch, 000h
00012E41 7030303030FC00      <1>
00012E48 78CC0C3860CCFC0078- <1>      db  078h, 0cch, 00ch, 038h, 060h, 0cch, 0fch, 000h, 078h, 0cch, 00ch, 038h, 00ch, 0cch, 078h, 000h
00012E51 CC0C380CCC7800      <1>
00012E58 1C3C6CCCFE0C1E00FC- <1>      db  01ch, 03ch, 06ch, 0cch, 0feh, 00ch, 01eh, 000h, 0fch, 0c0h, 0f8h, 00ch, 00ch, 0cch, 078h, 000h
00012E61 C0F80C0CCC7800      <1>
00012E68 3860C0F8CCCC7800FC- <1>      db  038h, 060h, 0c0h, 0f8h, 0cch, 0cch, 078h, 000h, 0fch, 0cch, 00ch, 018h, 030h, 030h, 030h, 000h
00012E71 CC0C1830303000      <1>
00012E78 78CCCC78CCCC780078- <1>      db  078h, 0cch, 0cch, 078h, 0cch, 0cch, 078h, 000h, 078h, 0cch, 0cch, 07ch, 00ch, 018h, 070h, 000h
00012E81 CCCC7C0C187000      <1>
00012E88 003030000030300000- <1>      db  000h, 030h, 030h, 000h, 000h, 030h, 030h, 000h, 000h, 030h, 030h, 000h, 000h, 030h, 030h, 060h
00012E91 30300000303060      <1>
00012E98 183060C06030180000- <1>      db  018h, 030h, 060h, 0c0h, 060h, 030h, 018h, 000h, 000h, 000h, 0fch, 000h, 000h, 0fch, 000h, 000h
00012EA1 00FC0000FC0000      <1>
00012EA8 6030180C1830600078- <1>      db  060h, 030h, 018h, 00ch, 018h, 030h, 060h, 000h, 078h, 0cch, 00ch, 018h, 030h, 000h, 030h, 000h
00012EB1 CC0C1830003000      <1>
00012EB8 7CC6DEDEDEC0780030- <1>      db  07ch, 0c6h, 0deh, 0deh, 0deh, 0c0h, 078h, 000h, 030h, 078h, 0cch, 0cch, 0fch, 0cch, 0cch, 000h
00012EC1 78CCCCFCCCCC00      <1>
00012EC8 FC66667C6666FC003C- <1>      db  0fch, 066h, 066h, 07ch, 066h, 066h, 0fch, 000h, 03ch, 066h, 0c0h, 0c0h, 0c0h, 066h, 03ch, 000h
00012ED1 66C0C0C0663C00      <1>
00012ED8 F86C6666666CF800FE- <1>      db  0f8h, 06ch, 066h, 066h, 066h, 06ch, 0f8h, 000h, 0feh, 062h, 068h, 078h, 068h, 062h, 0feh, 000h
00012EE1 6268786862FE00      <1>
00012EE8 FE6268786860F0003C- <1>      db  0feh, 062h, 068h, 078h, 068h, 060h, 0f0h, 000h, 03ch, 066h, 0c0h, 0c0h, 0ceh, 066h, 03eh, 000h
00012EF1 66C0C0CE663E00      <1>
00012EF8 CCCCCCFCCCCCCC0078- <1>      db  0cch, 0cch, 0cch, 0fch, 0cch, 0cch, 0cch, 000h, 078h, 030h, 030h, 030h, 030h, 030h, 078h, 000h
00012F01 30303030307800      <1>
00012F08 1E0C0C0CCCCC7800E6- <1>      db  01eh, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 000h, 0e6h, 066h, 06ch, 078h, 06ch, 066h, 0e6h, 000h
00012F11 666C786C66E600      <1>
00012F18 F06060606266FE00C6- <1>      db  0f0h, 060h, 060h, 060h, 062h, 066h, 0feh, 000h, 0c6h, 0eeh, 0feh, 0feh, 0d6h, 0c6h, 0c6h, 000h
00012F21 EEFEFED6C6C600      <1>
00012F28 C6E6F6DECEC6C60038- <1>      db  0c6h, 0e6h, 0f6h, 0deh, 0ceh, 0c6h, 0c6h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h
00012F31 6CC6C6C66C3800      <1>
00012F38 FC66667C6060F00078- <1>      db  0fch, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h, 078h, 0cch, 0cch, 0cch, 0dch, 078h, 01ch, 000h
00012F41 CCCCCCDC781C00      <1>
00012F48 FC66667C6C66E60078- <1>      db  0fch, 066h, 066h, 07ch, 06ch, 066h, 0e6h, 000h, 078h, 0cch, 0e0h, 070h, 01ch, 0cch, 078h, 000h
00012F51 CCE0701CCC7800      <1>
00012F58 FCB4303030307800CC- <1>      db  0fch, 0b4h, 030h, 030h, 030h, 030h, 078h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 0fch, 000h
00012F61 CCCCCCCCCCFC00      <1>
00012F68 CCCCCCCCCC783000C6- <1>      db  0cch, 0cch, 0cch, 0cch, 0cch, 078h, 030h, 000h, 0c6h, 0c6h, 0c6h, 0d6h, 0feh, 0eeh, 0c6h, 000h
00012F71 C6C6D6FEEEC600      <1>
00012F78 C6C66C38386CC600CC- <1>      db  0c6h, 0c6h, 06ch, 038h, 038h, 06ch, 0c6h, 000h, 0cch, 0cch, 0cch, 078h, 030h, 030h, 078h, 000h
00012F81 CCCC7830307800      <1>
00012F88 FEC68C183266FE0078- <1>      db  0feh, 0c6h, 08ch, 018h, 032h, 066h, 0feh, 000h, 078h, 060h, 060h, 060h, 060h, 060h, 078h, 000h
00012F91 60606060607800      <1>
00012F98 C06030180C06020078- <1>      db  0c0h, 060h, 030h, 018h, 00ch, 006h, 002h, 000h, 078h, 018h, 018h, 018h, 018h, 018h, 078h, 000h
00012FA1 18181818187800      <1>
00012FA8 10386CC60000000000- <1>      db  010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
00012FB1 000000000000FF      <1>
00012FB8 303018000000000000- <1>      db  030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 076h, 000h
00012FC1 00780C7CCC7600      <1>
00012FC8 E060607C6666DC0000- <1>      db  0e0h, 060h, 060h, 07ch, 066h, 066h, 0dch, 000h, 000h, 000h, 078h, 0cch, 0c0h, 0cch, 078h, 000h
00012FD1 0078CCC0CC7800      <1>
00012FD8 1C0C0C7CCCCC760000- <1>      db  01ch, 00ch, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h
00012FE1 0078CCFCC07800      <1>
00012FE8 386C60F06060F00000- <1>      db  038h, 06ch, 060h, 0f0h, 060h, 060h, 0f0h, 000h, 000h, 000h, 076h, 0cch, 0cch, 07ch, 00ch, 0f8h
00012FF1 0076CCCC7C0CF8      <1>
00012FF8 E0606C766666E60030- <1>      db  0e0h, 060h, 06ch, 076h, 066h, 066h, 0e6h, 000h, 030h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
00013001 00703030307800      <1>
00013008 0C000C0C0CCCCC78E0- <1>      db  00ch, 000h, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 0e0h, 060h, 066h, 06ch, 078h, 06ch, 0e6h, 000h
00013011 60666C786CE600      <1>
00013018 703030303030780000- <1>      db  070h, 030h, 030h, 030h, 030h, 030h, 078h, 000h, 000h, 000h, 0cch, 0feh, 0feh, 0d6h, 0c6h, 000h
00013021 00CCFEFED6C600      <1>
00013028 0000F8CCCCCCCC0000- <1>      db  000h, 000h, 0f8h, 0cch, 0cch, 0cch, 0cch, 000h, 000h, 000h, 078h, 0cch, 0cch, 0cch, 078h, 000h
00013031 0078CCCCCC7800      <1>
00013038 0000DC66667C60F000- <1>      db  000h, 000h, 0dch, 066h, 066h, 07ch, 060h, 0f0h, 000h, 000h, 076h, 0cch, 0cch, 07ch, 00ch, 01eh
00013041 0076CCCC7C0C1E      <1>
00013048 0000DC766660F00000- <1>      db  000h, 000h, 0dch, 076h, 066h, 060h, 0f0h, 000h, 000h, 000h, 07ch, 0c0h, 078h, 00ch, 0f8h, 000h
00013051 007CC0780CF800      <1>
00013058 10307C303034180000- <1>      db  010h, 030h, 07ch, 030h, 030h, 034h, 018h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 076h, 000h
00013061 00CCCCCCCC7600      <1>
00013068 0000CCCCCC78300000- <1>      db  000h, 000h, 0cch, 0cch, 0cch, 078h, 030h, 000h, 000h, 000h, 0c6h, 0d6h, 0feh, 0feh, 06ch, 000h
00013071 00C6D6FEFE6C00      <1>
00013078 0000C66C386CC60000- <1>      db  000h, 000h, 0c6h, 06ch, 038h, 06ch, 0c6h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 07ch, 00ch, 0f8h
00013081 00CCCCCC7C0CF8      <1>
00013088 0000FC983064FC001C- <1>      db  000h, 000h, 0fch, 098h, 030h, 064h, 0fch, 000h, 01ch, 030h, 030h, 0e0h, 030h, 030h, 01ch, 000h
00013091 3030E030301C00      <1>
00013098 181818001818180OE0- <1>      db  018h, 018h, 018h, 000h, 018h, 018h, 018h, 000h, 0e0h, 030h, 030h, 01ch, 030h, 030h, 0e0h, 000h
000130A1 30301C3030E000      <1>
000130A8 76DC00000000000000- <1>      db  076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 000h
000130B1 10386CC6C6FE00      <1>
000130B8 78CCC0CC78180C7800- <1>      db  078h, 0cch, 0c0h, 0cch, 078h, 018h, 00ch, 078h, 000h, 0cch, 000h, 0cch, 0cch, 0cch, 07eh, 000h
000130C1 CC00CCCCCC7E00      <1>
000130C8 1C0078CCFCC078007E- <1>      db  01ch, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h, 07eh, 0c3h, 03ch, 006h, 03eh, 066h, 03fh, 000h
000130D1 C33C063E663F00      <1>
000130D8 CC00780C7CCC7E00E0- <1>      db  0cch, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 0e0h, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h
000130E1 00780C7CCC7E00      <1>
000130E8 3030780C7CCC7E0000- <1>      db  030h, 030h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 000h, 000h, 078h, 0c0h, 0c0h, 078h, 00ch, 038h
000130F1 0078C0C0780C38      <1>
000130F8 7EC33C667E603C00CC- <1>      db  07eh, 0c3h, 03ch, 066h, 07eh, 060h, 03ch, 000h, 0cch, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h
00013101 0078CCFCC07800      <1>
00013108 E00078CCFCC07800CC- <1>      db  0e0h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h, 0cch, 000h, 070h, 030h, 030h, 030h, 078h, 000h
00013111 00703030307800      <1>
00013118 7CC6381818183C00E0- <1>      db  07ch, 0c6h, 038h, 018h, 018h, 018h, 03ch, 000h, 0e0h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
00013121 00703030307800      <1>
00013128 C6386CC6FEC6C60030- <1>      db  0c6h, 038h, 06ch, 0c6h, 0feh, 0c6h, 0c6h, 000h, 030h, 030h, 000h, 078h, 0cch, 0fch, 0cch, 000h
00013131 300078CCFCCC00      <1>
00013138 1C00FC607860FC0000- <1>      db  01ch, 000h, 0fch, 060h, 078h, 060h, 0fch, 000h, 000h, 000h, 07fh, 00ch, 07fh, 0cch, 07fh, 000h
00013141 007F0C7FCC7F00      <1>
00013148 3E6CCCFECCCCCE0078- <1>      db  03eh, 06ch, 0cch, 0feh, 0cch, 0cch, 0ceh, 000h, 078h, 0cch, 000h, 078h, 0cch, 0cch, 078h, 000h
00013151 CC0078CCCC7800      <1>
00013158 00CC0078CCCC780000- <1>      db  000h, 0cch, 000h, 078h, 0cch, 0cch, 078h, 000h, 000h, 0e0h, 000h, 078h, 0cch, 0cch, 078h, 000h
00013161 E00078CCCC7800      <1>
00013168 78CC00CCCCCC7E0000- <1>      db  078h, 0cch, 000h, 0cch, 0cch, 0cch, 07eh, 000h, 000h, 0e0h, 000h, 0cch, 0cch, 0cch, 07eh, 000h
00013171 E000CCCCCC7E00      <1>
00013178 00CC00CCCC7C0CF8C3- <1>      db  000h, 0cch, 000h, 0cch, 0cch, 07ch, 00ch, 0f8h, 0c3h, 018h, 03ch, 066h, 066h, 03ch, 018h, 000h
00013181 183C66663C1800      <1>
00013188 CC00CCCCCCCC780018- <1>      db  0cch, 000h, 0cch, 0cch, 0cch, 0cch, 078h, 000h, 018h, 018h, 07eh, 0c0h, 0c0h, 07eh, 018h, 018h
00013191 187EC0C07E1818      <1>
00013198 386C64F060E6FC00CC- <1>      db  038h, 06ch, 064h, 0f0h, 060h, 0e6h, 0fch, 000h, 0cch, 0cch, 078h, 0fch, 030h, 0fch, 030h, 030h
```

```
000131A1 CC78FC30FC3030      <1>
000131A8 F8CCCCFAC6CFC6C70E- <1>     db  0f8h, 0cch, 0cch, 0fah, 0c6h, 0cfh, 0c6h, 0c7h, 00eh, 01bh, 018h, 03ch, 018h, 018h, 0d8h, 070h
000131B1 1B183C1818D870      <1>
000131B8 1C00780C7CCC7E0038- <1>     db  01ch, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 038h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
000131C1 00703030307800      <1>
000131C8 001C0078CCCC780000- <1>     db  000h, 01ch, 000h, 078h, 0cch, 0cch, 078h, 000h, 000h, 01ch, 000h, 0cch, 0cch, 0cch, 07eh, 000h
000131D1 1C00CCCCCC7E00      <1>
000131D8 00F800F8CCCCCC00FC- <1>     db  000h, 0f8h, 000h, 0f8h, 0cch, 0cch, 0cch, 000h, 0fch, 000h, 0cch, 0ech, 0fch, 0dch, 0cch, 000h
000131E1 00CCECFCDCCC00      <1>
000131E8 3C6C6C3E007E000038- <1>     db  03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h
000131F1 6C6C38007C0000      <1>
000131F8 30003060C0CC780000- <1>     db  030h, 000h, 030h, 060h, 0c0h, 0cch, 078h, 000h, 000h, 000h, 000h, 0fch, 0c0h, 0c0h, 000h, 000h
00013201 0000FCC0C00000      <1>
00013208 000000FC0C0C0000C3- <1>     db  000h, 000h, 000h, 0fch, 00ch, 00ch, 000h, 000h, 0c3h, 0c6h, 0cch, 0deh, 033h, 066h, 0cch, 00fh
00013211 C6CCDE3366CC0F      <1>
00013218 C3C6CCDB376FCF0318- <1>     db  0c3h, 0c6h, 0cch, 0dbh, 037h, 06fh, 0cfh, 003h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 000h
00013221 18001818181800      <1>
00013228 003366CC6633000000- <1>     db  000h, 033h, 066h, 0cch, 066h, 033h, 000h, 000h, 000h, 0cch, 066h, 033h, 066h, 0cch, 000h, 000h
00013231 CC663366CC0000      <1>
00013238 228822882288228855- <1>     db  022h, 088h, 022h, 088h, 022h, 088h, 022h, 088h, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah
00013241 AA55AA55AA55AA      <1>
00013248 DB77DBEEDB77DBEE18- <1>     db  0dbh, 077h, 0dbh, 0eeh, 0dbh, 077h, 0dbh, 0eeh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
00013251 18181818181818      <1>
00013258 18181818F818181818- <1>     db  018h, 018h, 018h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h, 018h
00013261 18F818F8181818      <1>
00013268 36363636F636363600- <1>     db  036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h
00013271 000000FE363636      <1>
00013278 0000F818F818181836- <1>     db  000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h
00013281 36F606F6363636      <1>
00013288 363636363636363600- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 0feh, 006h, 0f6h, 036h, 036h, 036h
00013291 00FE06F6363636      <1>
00013298 3636F606FE00000036- <1>     db  036h, 036h, 0f6h, 006h, 0feh, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h, 000h
000132A1 363636FE000000      <1>
000132A8 1818F818F800000000- <1>     db  018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h
000132B1 000000F8181818      <1>
000132B8 181818181F00000018- <1>     db  018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h
000132C1 181818FF000000      <1>
000132C8 00000000FF18181818- <1>     db  000h, 000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 018h, 018h
000132D1 1818181F181818      <1>
000132D8 00000000FF00000018- <1>     db  000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h
000132E1 181818FF181818      <1>
000132E8 18181F181F18181836- <1>     db  018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h
000132F1 36363637363636      <1>
000132F8 363637303F00000000- <1>     db  036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h
00013301 003F3037363636      <1>
00013308 3636F700FF00000000- <1>     db  036h, 036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0f7h, 036h, 036h, 036h
00013311 00FF00F7363636      <1>
00013318 363637303736363600- <1>     db  036h, 036h, 037h, 030h, 037h, 036h, 036h, 036h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h, 000h
00013321 00FF00FF000000      <1>
00013328 3636F700F736363618- <1>     db  036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h
00013331 18FF00FF000000      <1>
00013338 36363636FF00000000- <1>     db  036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h
00013341 00FF00FF181818      <1>
00013348 00000000FF36363636- <1>     db  000h, 000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 03fh, 000h, 000h, 000h
00013351 3636363F000000      <1>
00013358 18181F181F00000000- <1>     db  018h, 018h, 01fh, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h, 018h
00013361 001F181F181818      <1>
00013368 000000003F36363636- <1>     db  000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h
00013371 363636FF363636      <1>
00013378 1818FF18FF18181818- <1>     db  018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h
00013381 181818F8000000      <1>
00013388 000000001F181818FF- <1>     db  000h, 000h, 000h, 000h, 01fh, 018h, 018h, 018h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
00013391 FFFFFFFFFFFFFF      <1>
00013398 00000000FFFFFFFFF0- <1>     db  000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h
000133A1 F0F0F0F0F0F0F0      <1>
000133A8 0F0F0F0F0F0F0F0FFF- <1>     db  00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h
000133B1 FFFFFF00000000      <1>
000133B8 000076DCC8DC760000- <1>     db  000h, 000h, 076h, 0dch, 0c8h, 0dch, 076h, 000h, 000h, 078h, 0cch, 0f8h, 0cch, 0f8h, 0c0h, 0c0h
000133C1 78CCF8CCF8C0C0      <1>
000133C8 00FCCCC0C0C0C00000- <1>     db  000h, 0fch, 0cch, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 0feh, 06ch, 06ch, 06ch, 06ch, 06ch, 000h
000133D1 FE6C6C6C6C6C00      <1>
000133D8 FCCC603060CCFC0000- <1>     db  0fch, 0cch, 060h, 030h, 060h, 0cch, 0fch, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 070h, 000h
000133E1 007ED8D8D87000      <1>
000133E8 00666666667C60C000- <1>     db  000h, 066h, 066h, 066h, 066h, 07ch, 060h, 0c0h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 000h
000133F1 76DC1818181800      <1>
000133F8 FC3078CCCC7830FC38- <1>     db  0fch, 030h, 078h, 0cch, 0cch, 078h, 030h, 0fch, 038h, 06ch, 0c6h, 0feh, 0c6h, 06ch, 038h, 000h
00013401 6CC6FEC66C3800      <1>
00013408 386CC6C66C6CEE001C- <1>     db  038h, 06ch, 0c6h, 0c6h, 06ch, 06ch, 0eeh, 000h, 01ch, 030h, 018h, 07ch, 0cch, 0cch, 078h, 000h
00013411 30187CCCCC7800      <1>
00013418 00007EDBDB7E000006- <1>     db  000h, 000h, 07eh, 0dbh, 0dbh, 07eh, 000h, 000h, 006h, 00ch, 07eh, 0dbh, 0dbh, 07eh, 060h, 0c0h
00013421 0C7EDBDB7E60C0      <1>
00013428 3860C0F8C060380078- <1>     db  038h, 060h, 0c0h, 0f8h, 0c0h, 060h, 038h, 000h, 078h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 000h
00013431 CCCCCCCCCCCC00      <1>
00013438 00FC00FC00FC000030- <1>     db  000h, 0fch, 000h, 0fch, 000h, 0fch, 000h, 000h, 030h, 030h, 0fch, 030h, 030h, 000h, 0fch, 000h
00013441 30FC303000FC00      <1>
00013448 603018306000FC0018- <1>     db  060h, 030h, 018h, 030h, 060h, 000h, 0fch, 000h, 018h, 030h, 060h, 030h, 018h, 000h, 0fch, 000h
00013451 3060301800FC00      <1>
00013458 0E1B1B181818181818- <1>     db  00eh, 01bh, 01bh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h, 070h
00013461 18181818D8D870      <1>
00013468 303000FC0030300000- <1>     db  030h, 030h, 000h, 0fch, 000h, 030h, 030h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h
00013471 76DC0076DC0000      <1>
00013478 386C6C380000000000- <1>     db  038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h
00013481 00001818000000      <1>
00013488 00000000180000000F- <1>     db  000h, 000h, 000h, 000h, 018h, 000h, 000h, 000h, 00fh, 00ch, 00ch, 00ch, 0ech, 06ch, 03ch, 01ch
00013491 0C0C0CEC6C3C1C      <1>
00013498 786C6C6C6C00000070- <1>     db  078h, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 070h, 018h, 030h, 060h, 078h, 000h, 000h, 000h
000134A1 18306078000000      <1>
000134A8 00003C3C3C3C000000- <1>     db  000h, 000h, 03ch, 03ch, 03ch, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
000134B1 00000000000000      <1>
                             <1> vgafont14:
000134B8 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
000134C1 00000000000000      <1>
000134C8 7E81A58181BD99817E- <1>     db  07eh, 081h, 0a5h, 081h, 081h, 0bdh, 099h, 081h, 07eh, 000h, 000h, 000h, 000h, 000h, 07eh, 0ffh
000134D1 00000000007EFF      <1>
000134D8 DBFFFFC3E7FF7E0000- <1>     db  0dbh, 0ffh, 0ffh, 0c3h, 0e7h, 0ffh, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 06ch, 0feh, 0feh
000134E1 000000006CFEFE      <1>
000134E8 FEFE7C381000000000- <1>     db  0feh, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 07ch, 0feh, 07ch
000134F1 000010387CFE7C      <1>
000134F8 381000000000000018- <1>     db  038h, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 0e7h, 0e7h, 0e7h, 018h, 018h
```

```
255 00013501 3C3CE7E7E71818      <1>
256 00013508 3C0000000000183C7E- <1>     db  03ch, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 07eh, 018h, 018h, 03ch, 000h
256 00013511 FFFF7E18183C00      <1>
257 00013518 0000000000000183C- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h, 000h
257 00013521 3C180000000000      <1>
258 00013528 FFFFFFFFFFE7C3C3E7- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h
258 00013531 FFFFFFFFFF0000      <1>
259 00013538 00003C664242663C00- <1>     db  000h, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh
259 00013541 000000FFFFFFFF      <1>
260 00013548 C399BDBD99C3FFFFFF- <1>     db  0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 01eh, 00eh, 01ah, 032h
260 00013551 FF00001E0E1A32      <1>
261 00013558 78CCCCCC7800000000- <1>     db  078h, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 066h, 066h, 03ch, 018h
261 00013561 003C6666663C18      <1>
262 00013568 7E181800000000003F- <1>     db  07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 03fh, 033h, 03fh, 030h, 030h, 030h, 070h, 0f0h
262 00013571 333F30303070F0      <1>
263 00013578 E000000000007F637F- <1>     db  0e0h, 000h, 000h, 000h, 000h, 000h, 07fh, 063h, 07fh, 063h, 063h, 063h, 067h, 0e7h, 0e6h, 0c0h
263 00013581 63636367E7E6C0      <1>
264 00013588 000000001818DB3CE7- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 0dbh, 03ch, 0e7h, 03ch, 0dbh, 018h, 018h, 000h, 000h, 000h
264 00013591 3CDB1818000000      <1>
265 00013598 000080C0E0F8FEF8E0- <1>     db  000h, 000h, 080h, 0c0h, 0e0h, 0f8h, 0feh, 0f8h, 0e0h, 0c0h, 080h, 000h, 000h, 000h, 000h, 000h
265 000135A1 C0800000000000      <1>
266 000135A8 02060E3EFE3E0E0602- <1>     db  002h, 006h, 00eh, 03eh, 0feh, 03eh, 00eh, 006h, 002h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch
266 000135B1 0000000000183C      <1>
267 000135B8 7E1818187E3C180000- <1>     db  07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h
267 000135C1 00000066666666      <1>
268 000135C8 666600666600000000- <1>     db  066h, 066h, 000h, 066h, 066h, 000h, 000h, 000h, 000h, 000h, 07fh, 0dbh, 0dbh, 0dbh, 07bh, 01bh
268 000135D1 007FDBDBDB7B1B      <1>
269 000135D8 1B1B1B000000007CC6- <1>     db  01bh, 01bh, 01bh, 000h, 000h, 000h, 000h, 07ch, 0c6h, 060h, 038h, 06ch, 0c6h, 0c6h, 06ch, 038h
269 000135E1 60386CC6C66C38      <1>
270 000135E8 0CC67C000000000000- <1>     db  00ch, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 0feh, 000h
270 000135F1 000000FEFEFE00      <1>
271 000135F8 00000000183C7E1818- <1>     db  000h, 000h, 000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 07eh, 000h, 000h
271 00013601 187E3C187E0000      <1>
272 00013608 0000183C7E18181818- <1>     db  000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h
272 00013611 18180000000000      <1>
273 00013618 181818181818187E3C18- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
273 00013621 00000000000000      <1>
274 00013628 180CFE0C1800000000- <1>     db  018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 030h, 060h
274 00013631 00000000003060      <1>
275 00013638 FE6030000000000000- <1>     db  0feh, 060h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c0h
275 00013641 00000000C0C0C0      <1>
276 00013648 FE0000000000000000- <1>     db  0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 028h, 06ch, 0feh, 06ch, 028h, 000h
276 00013651 00286CFE6C2800      <1>
277 00013658 000000000000001038- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 038h, 07ch, 07ch, 0feh, 0feh, 000h, 000h
277 00013661 387C7CFEFE0000      <1>
278 00013668 0000000000FEFE7C7C- <1>     db  000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 07ch, 07ch, 038h, 038h, 010h, 000h, 000h, 000h, 000h
278 00013671 38381000000000      <1>
279 00013678 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
279 00013681 00000000000000      <1>
280 00013688 183C3C3C1818001818- <1>     db  018h, 03ch, 03ch, 03ch, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 066h, 066h, 066h
280 00013691 00000000666666      <1>
281 00013698 240000000000000000- <1>     db  024h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch
281 000136A1 0000006C6CFE6C      <1>
282 000136A8 6C6CFE6C6C00000018- <1>     db  06ch, 06ch, 0feh, 06ch, 06ch, 000h, 000h, 000h, 018h, 018h, 07ch, 0c6h, 0c2h, 0c0h, 07ch, 006h
282 000136B1 187CC6C2C07C06      <1>
283 000136B8 86C67C181800000000- <1>     db  086h, 0c6h, 07ch, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 0c2h, 0c6h, 00ch, 018h, 030h, 066h
283 000136C1 00C2C60C183066      <1>
284 000136C8 C60000000000386C6C- <1>     db  0c6h, 000h, 000h, 000h, 000h, 000h, 038h, 06ch, 06ch, 038h, 076h, 0dch, 0cch, 0cch, 076h, 000h
284 000136D1 3876DCCCCC7600      <1>
285 000136D8 000000303030600000- <1>     db  000h, 000h, 000h, 030h, 030h, 030h, 060h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
285 000136E1 00000000000000      <1>
286 000136E8 00000C183030303030- <1>     db  000h, 000h, 00ch, 018h, 030h, 030h, 030h, 030h, 030h, 018h, 00ch, 000h, 000h, 000h, 000h, 000h
286 000136F1 180C0000000000      <1>
287 000136F8 30180C0C0C0C0C1830- <1>     db  030h, 018h, 00ch, 00ch, 00ch, 00ch, 00ch, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
287 00013701 00000000000000      <1>
288 00013708 663CFF3C6600000000- <1>     db  066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h
288 00013711 00000000001818      <1>
289 00013718 7E1818000000000000- <1>     db  07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
289 00013721 00000000000000      <1>
290 00013728 181818300000000000- <1>     db  018h, 018h, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 000h, 000h, 000h
290 00013731 000000FE000000      <1>
291 00013738 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h
291 00013741 00000000181800      <1>
292 00013748 0000000002060C1830- <1>     db  000h, 000h, 000h, 000h, 002h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h, 000h, 000h, 000h
292 00013751 60C08000000000      <1>
293 00013758 00007CC6CEDEF6E6C6- <1>     db  000h, 000h, 07ch, 0c6h, 0ceh, 0deh, 0f6h, 0e6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
293 00013761 C67C0000000000      <1>
294 00013768 183878181818181818187E- <1>     db  018h, 038h, 078h, 018h, 018h, 018h, 018h, 018h, 07eh, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h
294 00013771 00000000007CC6      <1>
295 00013778 060C183060C6FE0000- <1>     db  006h, 00ch, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 006h, 006h
295 00013781 0000007CC60606      <1>
296 00013788 3C0606C67C00000000- <1>     db  03ch, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 00ch, 01ch, 03ch, 06ch, 0cch, 0feh
296 00013791 000C1C3C6CCCFE      <1>
297 00013798 0C0C1E0000000000FE- <1>     db  00ch, 00ch, 01eh, 000h, 000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0fch, 006h, 006h, 0c6h
297 000137A1 C0C0C0FC0606C6      <1>
298 000137A8 7C00000000003860C0- <1>     db  07ch, 000h, 000h, 000h, 000h, 000h, 038h, 060h, 0c0h, 0c0h, 0fch, 0c6h, 0c6h, 0c6h, 07ch, 000h
298 000137B1 C0FCC6C6C67C00      <1>
299 000137B8 00000000FEC6060C18- <1>     db  000h, 000h, 000h, 000h, 0feh, 0c6h, 006h, 00ch, 018h, 030h, 030h, 030h, 030h, 000h, 000h, 000h
299 000137C1 30303030000000      <1>
300 000137C8 00007CC6C6C67CC6C6- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
300 000137D1 C67C0000000000      <1>
301 000137D8 7CC6C6C67E06060C78- <1>     db  07ch, 0c6h, 0c6h, 0c6h, 07eh, 006h, 006h, 00ch, 078h, 000h, 000h, 000h, 000h, 000h, 000h, 018h
301 000137E1 00000000000018      <1>
302 000137E8 180000001818000000- <1>     db  018h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h
302 000137F1 00000000181800      <1>
303 000137F8 000018183000000000- <1>     db  000h, 000h, 018h, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 006h, 00ch, 018h, 030h, 060h, 030h
303 00013801 00060C18306030      <1>
304 00013808 180C06000000000000- <1>     db  018h, 00ch, 006h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 000h, 000h, 07eh, 000h
304 00013811 00007E00007E00      <1>
305 00013818 000000000000603018- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 060h, 000h
305 00013821 0C060C18306000      <1>
306 00013828 000000007CC6C60C18- <1>     db  000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 00ch, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h
306 00013831 18001818000000      <1>
307 00013838 00007CC6C6DEDEDEDC- <1>     db  000h, 000h, 07ch, 0c6h, 0c6h, 0deh, 0deh, 0deh, 0dch, 0c0h, 07ch, 000h, 000h, 000h, 000h, 000h
307 00013841 C07C0000000000      <1>
308 00013848 10386CC6C6FEC6C6C6- <1>     db  010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h, 0fch, 066h
308 00013851 0000000000FC66      <1>
309 00013858 66667C666666FC0000- <1>     db  066h, 066h, 07ch, 066h, 066h, 066h, 0fch, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 0c2h, 0c0h
309 00013861 0000003C66C2C0      <1>
```

```
310 00013868 C0C0C2663C00000000- <1>     db  0c0h, 0c0h, 0c2h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h, 0f8h, 06ch, 066h, 066h, 066h, 066h
310 00013871 00F86C66666666      <1>
311 00013878 666CF80000000000FE- <1>     db  066h, 06ch, 0f8h, 000h, 000h, 000h, 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 062h, 066h
311 00013881 66626878686266      <1>
312 00013888 FE0000000000FE6662- <1>     db  0feh, 000h, 000h, 000h, 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 060h, 0f0h, 000h
312 00013891 6878686060F000      <1>
313 00013898 000000003C66C2C0C0- <1>     db  000h, 000h, 000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0deh, 0c6h, 066h, 03ah, 000h, 000h, 000h
313 000138A1 DEC6663A000000      <1>
314 000138A8 0000C6C6C6C6FEC6C6- <1>     db  000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h
314 000138B1 C6C60000000000      <1>
315 000138B8 3C181818181818183C- <1>     db  03ch, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 01eh, 00ch
315 000138C1 00000000001E0C      <1>
316 000138C8 0C0C0C0CCCCC780000- <1>     db  00ch, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h, 000h, 0e6h, 066h, 06ch, 06ch
316 000138D1 000000E6666C6C      <1>
317 000138D8 786C6C666E60000000- <1>     db  078h, 06ch, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 0f0h, 060h, 060h, 060h, 060h, 060h
317 000138E1 00F06060606060      <1>
318 000138E8 6266FE0000000000C6- <1>     db  062h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h, 0c6h, 0eeh, 0feh, 0feh, 0d6h, 0c6h, 0c6h, 0c6h
318 000138F1 EEFEFED6C6C6C6      <1>
319 000138F8 C60000000000C6E6F6- <1>     db  0c6h, 000h, 000h, 000h, 000h, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 000h
319 00013901 FEDECEC6C6C600      <1>
320 00013908 00000000386CC6C6C6- <1>     db  000h, 000h, 000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h
320 00013911 C6C66C38000000      <1>
321 00013918 0000FC6666667C6060- <1>     db  000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h, 000h
321 00013921 60F00000000000      <1>
322 00013928 7CC6C6C6C6D6DE7C0C- <1>     db  07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0deh, 07ch, 00ch, 00eh, 000h, 000h, 000h, 000h, 0fch, 066h
322 00013931 0E00000000FC66      <1>
323 00013938 66667C6C6666E60000- <1>     db  066h, 066h, 07ch, 06ch, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 060h
323 00013941 0000007CC6C660      <1>
324 00013948 380CC6C67C00000000- <1>     db  038h, 00ch, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 07eh, 07eh, 05ah, 018h, 018h, 018h
324 00013951 007E7E5A181818      <1>
325 00013958 18183C0000000000C6- <1>     db  018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h
325 00013961 C6C6C6C6C6C6C6      <1>
326 00013968 7C0000000000C6C6C6- <1>     db  07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 010h, 000h
326 00013971 C6C6C66C381000      <1>
327 00013978 00000000C6C6C6C6D6- <1>     db  000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0d6h, 0feh, 07ch, 06ch, 000h, 000h, 000h
327 00013981 D6FE7C6C000000      <1>
328 00013988 0000C6C66C3838386C- <1>     db  000h, 000h, 0c6h, 0c6h, 06ch, 038h, 038h, 038h, 06ch, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h
328 00013991 C6C60000000000      <1>
329 00013998 666666663C1818183C- <1>     db  066h, 066h, 066h, 066h, 03ch, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 0feh, 0c6h
329 000139A1 0000000000FEC6      <1>
330 000139A8 8C183060C2C6FE0000- <1>     db  08ch, 018h, 030h, 060h, 0c2h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 03ch, 030h, 030h, 030h
330 000139B1 0000003C303030      <1>
331 000139B8 303030303C00000000- <1>     db  030h, 030h, 030h, 030h, 03ch, 000h, 000h, 000h, 000h, 000h, 080h, 0c0h, 0e0h, 070h, 038h, 01ch
331 000139C1 0080C0E070381C      <1>
332 000139C8 0E060200000000003C- <1>     db  00eh, 006h, 002h, 000h, 000h, 000h, 000h, 000h, 03ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch
332 000139D1 0C0C0C0C0C0C0C      <1>
333 000139D8 3C00000010386CC600- <1>     db  03ch, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
333 000139E1 00000000000000      <1>
334 000139E8 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h
334 000139F1 0000000000FF00      <1>
335 000139F8 303018000000000000- <1>     db  030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
335 00013A01 00000000000000      <1>
336 00013A08 000000780C7CCCCC76- <1>     db  000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 0e0h, 060h
336 00013A11 0000000000E060      <1>
337 00013A18 60786C6666667C0000- <1>     db  060h, 078h, 06ch, 066h, 066h, 066h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch
337 00013A21 0000000000007C      <1>
338 00013A28 C6C0C0C67C00000000- <1>     db  0c6h, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 01ch, 00ch, 00ch, 03ch, 06ch, 0cch
338 00013A31 001C0C0C3C6CCC      <1>
339 00013A38 CCCC76000000000000- <1>     db  0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h
339 00013A41 00007CC6FEC0C6      <1>
340 00013A48 7C0000000000386C64- <1>     db  07ch, 000h, 000h, 000h, 000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 0f0h, 000h
340 00013A51 60F0606060F000      <1>
341 00013A58 000000000000076CC- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 07ch, 00ch, 0cch, 078h, 000h
341 00013A61 CCCC7C0CCC7800      <1>
342 00013A68 0000E060606C766666- <1>     db  000h, 000h, 0e0h, 060h, 060h, 06ch, 076h, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h
342 00013A71 66E60000000000      <1>
343 00013A78 18180038181818183C- <1>     db  018h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 006h, 006h
343 00013A81 00000000000606      <1>
344 00013A88 000E0606060666663C- <1>     db  000h, 00eh, 006h, 006h, 006h, 006h, 066h, 066h, 03ch, 000h, 000h, 000h, 0e0h, 060h, 060h, 066h
344 00013A91 000000E0606066      <1>
345 00013A98 6C786C66E600000000- <1>     db  06ch, 078h, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h
345 00013AA1 00381818181818      <1>
346 00013AA8 18183C000000000000- <1>     db  018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ech, 0feh, 0d6h, 0d6h, 0d6h
346 00013AB1 0000ECFED6D6D6      <1>
347 00013AB8 C60000000000000000- <1>     db  0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 000h
347 00013AC1 DC666666666600      <1>
348 00013AC8 000000000000007CC6- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h
348 00013AD1 C6C6C67C000000      <1>
349 00013AD8 0000000000DC666666- <1>     db  000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h, 000h, 000h
349 00013AE1 7C6060F0000000      <1>
350 00013AE8 00000076CCCCCC7C0C- <1>     db  000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 07ch, 00ch, 00ch, 01eh, 000h, 000h, 000h, 000h, 000h
350 00013AF1 0C1E0000000000      <1>
351 00013AF8 00DC76666060F00000- <1>     db  000h, 0dch, 076h, 066h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch
351 00013B01 0000000000007C      <1>
352 00013B08 C6701CC67C00000000- <1>     db  0c6h, 070h, 01ch, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 010h, 030h, 030h, 0fch, 030h, 030h
352 00013B11 00103030FC3030      <1>
353 00013B18 30361C000000000000- <1>     db  030h, 036h, 01ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch
353 00013B21 0000CCCCCCCCCC      <1>
354 00013B28 760000000000000000- <1>     db  076h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 03ch, 018h, 000h
354 00013B31 666666663C1800      <1>
355 00013B38 00000000000000C6C6- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0d6h, 0d6h, 0feh, 06ch, 000h, 000h, 000h
355 00013B41 D6D6FE6C000000      <1>
356 00013B48 0000000000C66C3838- <1>     db  000h, 000h, 000h, 000h, 000h, 0c6h, 06ch, 038h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h
356 00013B51 6CC60000000000      <1>
357 00013B58 000000C6C6C6C67E06- <1>     db  000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 0f8h, 000h, 000h, 000h, 000h, 000h
357 00013B61 0CF80000000000      <1>
358 00013B68 00FECC183066FE0000- <1>     db  000h, 0feh, 0cch, 018h, 030h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h, 00eh, 018h, 018h, 018h
358 00013B71 0000000E181818      <1>
359 00013B78 701818180E00000000- <1>     db  070h, 018h, 018h, 018h, 00eh, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 000h, 018h
359 00013B81 00181818180018      <1>
360 00013B88 181818000000000070- <1>     db  018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 070h, 018h, 018h, 018h, 00eh, 018h, 018h, 018h
360 00013B91 1818180E181818      <1>
361 00013B98 70000000000076DC00- <1>     db  070h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
361 00013BA1 00000000000000      <1>
362 00013BA8 00000000000010386C- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 000h, 000h, 000h, 000h
362 00013BB1 C6C6FE00000000      <1>
363 00013BB8 00003C66C2C0C0C266- <1>     db  000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 00ch, 006h, 07ch, 000h, 000h, 000h
363 00013BC1 3C0C067C000000      <1>
364 00013BC8 CCCC00CCCCCCCCCC76- <1>     db  0cch, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 00ch, 018h, 030h
```

```
364 00013BD1 000000000C1830       <1>
365 00013BD8 007CC6FEC0C67C0000- <1>     db  000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 078h
365 00013BE1 000010386C0078       <1>
366 00013BE8 0C7CCCCC7600000000- <1>     db  00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 000h, 078h, 00ch, 07ch
366 00013BF1 00CCCC00780C7C       <1>
367 00013BF8 CCCC76000000006030- <1>     db  0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 078h, 00ch, 07ch, 0cch, 0cch
367 00013C01 1800780C7CCCCC       <1>
368 00013C08 7600000000386C3800- <1>     db  076h, 000h, 000h, 000h, 000h, 038h, 06ch, 038h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h
368 00013C11 780C7CCCCC7600       <1>
369 00013C18 0000000000003C6660- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 060h, 066h, 03ch, 00ch, 006h, 03ch, 000h, 000h
369 00013C21 663C0C063C0000       <1>
370 00013C28 0010386C007CC6FEC0- <1>     db  000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
370 00013C31 C67C0000000000       <1>
371 00013C38 CCCC007CC6FEC0C67C- <1>     db  0cch, 0cch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h
371 00013C41 00000000603018       <1>
372 00013C48 007CC6FEC0C67C0000- <1>     db  000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 000h, 038h
372 00013C51 00000066660038       <1>
373 00013C58 181818183C00000000- <1>     db  018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 018h, 03ch, 066h, 000h, 038h, 018h, 018h
373 00013C61 183C6600381818       <1>
374 00013C68 18183C000000006030- <1>     db  018h, 018h, 03ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 038h, 018h, 018h, 018h, 018h
374 00013C71 18003818181818       <1>
375 00013C78 3C00000000C6C61038- <1>     db  03ch, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 000h
375 00013C81 6CC6C6FEC6C600       <1>
376 00013C88 0000386C3800386CC6- <1>     db  000h, 000h, 038h, 06ch, 038h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 000h, 000h, 000h
376 00013C91 C6FEC6C6000000       <1>
377 00013C98 18306000FE66607C60- <1>     db  018h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h
377 00013CA1 66FE0000000000       <1>
378 00013CA8 0000CC76367ED8D86E- <1>     db  000h, 000h, 0cch, 076h, 036h, 07eh, 0d8h, 0d8h, 06eh, 000h, 000h, 000h, 000h, 000h, 03eh, 06ch
378 00013CB1 00000000003E6C       <1>
379 00013CB8 CCCCFECCCCCCCE0000- <1>     db  0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 07ch
379 00013CC1 000010386C007C       <1>
380 00013CC8 C6C6C6C67C00000000- <1>     db  0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h, 07ch, 0c6h, 0c6h
380 00013CD1 00C6C6007CC6C6       <1>
381 00013CD8 C6C67C000000006030- <1>     db  0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h
381 00013CE1 18007CC6C6C6C6       <1>
382 00013CE8 7C000000003078CC00- <1>     db  07ch, 000h, 000h, 000h, 000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h
382 00013CF1 CCCCCCCCCC7600       <1>
383 00013CF8 00000060301800CCCC- <1>     db  000h, 000h, 000h, 060h, 030h, 018h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h
383 00013D01 CCCCCC76000000       <1>
384 00013D08 0000C6C600C6C6C6C6- <1>     db  000h, 000h, 0c6h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h, 000h, 0c6h
384 00013D11 7E060C780000C6       <1>
385 00013D18 C6386CC6C6C6C66C38- <1>     db  0c6h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h
385 00013D21 00000000C6C600       <1>
386 00013D28 C6C6C6C6C6C67C0000- <1>     db  0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 018h, 018h, 03ch, 066h, 060h
386 00013D31 000018183C6660       <1>
387 00013D38 60663C181800000000- <1>     db  060h, 066h, 03ch, 018h, 018h, 000h, 000h, 000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h
387 00013D41 386C6460F06060       <1>
388 00013D48 60E6FC000000000066- <1>     db  060h, 0e6h, 0fch, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 03ch, 018h, 07eh, 018h, 07eh, 018h
388 00013D51 663C187E187E18       <1>
389 00013D58 1800000000F8CCCCF8- <1>     db  018h, 000h, 000h, 000h, 000h, 0f8h, 0cch, 0cch, 0f8h, 0c4h, 0cch, 0deh, 0cch, 0cch, 0c6h, 000h
389 00013D61 C4CCDECCCCC600       <1>
390 00013D68 0000000E1B1818187E- <1>     db  000h, 000h, 000h, 00eh, 01bh, 018h, 018h, 018h, 07eh, 018h, 018h, 018h, 018h, 0d8h, 070h, 000h
390 00013D71 18181818D87000       <1>
391 00013D78 0018306000780C7CCC- <1>     db  000h, 018h, 030h, 060h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 00ch
391 00013D81 CC76000000000C       <1>
392 00013D88 1830003818181818 3C- <1>     db  018h, 030h, 000h, 038h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 018h, 030h, 060h
392 00013D91 00000000183060       <1>
393 00013D98 007CC6C6C6C67C0000- <1>     db  000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 018h, 030h, 060h, 000h, 0cch
393 00013DA1 000018306000CC       <1>
394 00013DA8 CCCCCCCC7600000000- <1>     db  0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 0dch, 066h, 066h
394 00013DB1 0076DC00DC6666       <1>
395 00013DB8 666666000000076DC00- <1>     db  066h, 066h, 066h, 000h, 000h, 000h, 076h, 0dch, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h
395 00013DC1 C6E6F6FEDECEC6       <1>
396 00013DC8 C6000000003C6C6C3E- <1>     db  0c6h, 000h, 000h, 000h, 000h, 03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 000h, 000h, 000h
396 00013DD1 007E0000000000       <1>
397 00013DD8 000000386C6C38007C- <1>     db  000h, 000h, 000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h
397 00013DE1 00000000000000       <1>
398 00013DE8 0000303000303060C6- <1>     db  000h, 000h, 030h, 030h, 000h, 030h, 030h, 060h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
398 00013DF1 C67C0000000000       <1>
399 00013DF8 00000000FEC0C0C000- <1>     db  000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
399 00013E01 00000000000000       <1>
400 00013E08 0000FE060606000000- <1>     db  000h, 000h, 0feh, 006h, 006h, 006h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c6h, 0cch, 0d8h
400 00013E11 0000C0C0C6CCD8       <1>
401 00013E18 3060DC860C183E0000- <1>     db  030h, 060h, 0dch, 086h, 00ch, 018h, 03eh, 000h, 000h, 0c0h, 0c0h, 0c6h, 0cch, 0d8h, 030h, 066h
401 00013E21 C0C0C6CCD83066       <1>
402 00013E28 CE9E3E060600000018- <1>     db  0ceh, 09eh, 03eh, 006h, 006h, 000h, 000h, 000h, 018h, 018h, 000h, 018h, 018h, 03ch, 03ch, 03ch
402 00013E31 180018183C3C3C       <1>
403 00013E38 180000000000000036- <1>     db  018h, 000h, 000h, 000h, 000h, 000h, 000h, 036h, 06ch, 0d8h, 06ch, 036h, 000h, 000h, 000h, 000h
403 00013E41 6CD86C36000000       <1>
404 00013E48 000000000000D86C36- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 0d8h, 06ch, 036h, 06ch, 0d8h, 000h, 000h, 000h, 000h, 000h
404 00013E51 6CD80000000000       <1>
405 00013E58 114411441144114411- <1>     db  011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 055h, 0aah
405 00013E61 441144114455AA       <1>
406 00013E68 55AA55AA55AA55AA55- <1>     db  055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 0ddh, 077h, 0ddh, 077h
406 00013E71 AA55AADD77DD77       <1>
407 00013E78 DD77DD77DD77DD77DD- <1>     db  0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 018h, 018h, 018h, 018h, 018h, 018h
407 00013E81 77181818181818       <1>
408 00013E88 181818181818181818- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h
408 00013E91 181818181818F8       <1>
409 00013E98 181818181818181818- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h
409 00013EA1 1818F818F81818       <1>
410 00013EA8 181818183636363636- <1>     db  018h, 018h, 018h, 018h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 036h
410 00013EB1 3636F636363636       <1>
411 00013EB8 363600000000000000- <1>     db  036h, 036h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h, 036h, 036h, 036h
411 00013EC1 FE363636363636       <1>
412 00013EC8 0000000000F818F818- <1>     db  000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 036h, 036h
412 00013ED1 18181818183636       <1>
413 00013ED8 363636F606F6363636- <1>     db  036h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
413 00013EE1 36363636363636       <1>
414 00013EE8 363636363636363636- <1>     db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 000h, 0feh
414 00013EF1 360000000000FE       <1>
415 00013EF8 06F636363636363636- <1>     db  006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0feh
415 00013F01 36363636F606FE       <1>
416 00013F08 000000000000363636- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h
416 00013F11 36363636FE0000       <1>
417 00013F18 000000001818181818- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h
417 00013F21 F818F800000000       <1>
418 00013F28 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h
418 00013F31 F8181818181818       <1>
```

```
419 00013F38 181818181818181F00- <1>      db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h
419 00013F41 00000000001818      <1>
420 00013F48 1818181818FF000000- <1>      db  018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
420 00013F51 00000000000000      <1>
421 00013F58 000000FF1818181818- <1>      db  000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
421 00013F61 18181818181818      <1>
422 00013F68 181F18181818181800- <1>      db  018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
422 00013F71 000000000000FF      <1>
423 00013F78 000000000000181818- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h
423 00013F81 18181818FF1818      <1>
424 00013F88 181818181818181818- <1>      db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h
424 00013F91 1F181F18181818      <1>
425 00013F98 181836363636363636- <1>      db  018h, 018h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h, 036h, 036h, 036h
425 00013FA1 37363636363636      <1>
426 00013FA8 363636363637303F00- <1>      db  036h, 036h, 036h, 036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
426 00013FB1 00000000000000      <1>
427 00013FB8 0000003F3037363636- <1>      db  000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
427 00013FC1 36363636363636      <1>
428 00013FC8 36F700FF0000000000- <1>      db  036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
428 00013FD1 000000000000FF      <1>
429 00013FD8 00F736363636363636- <1>      db  000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 030h, 037h
429 00013FE1 36363636373037      <1>
430 00013FE8 363636363636000000- <1>      db  036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h
430 00013FF1 0000FF00FF0000      <1>
431 00013FF8 000000003636363636- <1>      db  000h, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 036h
431 00014001 F700F736363636      <1>
432 00014008 3636181818181818FF00- <1>      db  036h, 036h, 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h
432 00014011 FF000000000000      <1>
433 00014018 36363636363636FF00- <1>      db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
433 00014021 00000000000000      <1>
434 00014028 000000FF00FF181818- <1>      db  000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
434 00014031 18181800000000      <1>
435 00014038 000000FF3636363636- <1>      db  000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
435 00014041 36363636363636      <1>
436 00014048 363F00000000000018- <1>      db  036h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh
436 00014051 181818181F181F      <1>
437 00014058 000000000000000000- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h
437 00014061 00001F181F1818      <1>
438 00014068 181818180000000000- <1>      db  018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h
438 00014071 00003F36363636      <1>
439 00014078 363636363636363636- <1>      db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h
439 00014081 FF363636363636      <1>
440 00014088 1818181818FF18FF18- <1>      db  018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
440 00014091 18181818181818      <1>
441 00014098 1818181818F8000000- <1>      db  018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
441 000140A1 00000000000000      <1>
442 000140A8 0000001F1818181818- <1>      db  000h, 000h, 000h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
442 000140B1 18FFFFFFFFFFFF      <1>
443 000140B8 FFFFFFFFFFFFFFFF00- <1>      db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
443 000140C1 000000000000FF      <1>
444 000140C8 FFFFFFFFFFFFF0F0F0- <1>      db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h
444 000140D1 F0F0F0F0F0F0F0      <1>
445 000140D8 F0F0F0F00F0F0F0F0F- <1>      db  0f0h, 0f0h, 0f0h, 0f0h, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh
445 000140E1 0F0F0F0F0F0F0F      <1>
446 000140E8 0F0FFFFFFFFFFFFFFF- <1>      db  00fh, 00fh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h
446 000140F1 00000000000000      <1>
447 000140F8 000000000076DCD8D8- <1>      db  000h, 000h, 000h, 000h, 000h, 076h, 0dch, 0d8h, 0d8h, 0dch, 076h, 000h, 000h, 000h, 000h, 000h
447 00014101 DC760000000000      <1>
448 00014108 00007CC6FCC6C6FCC0- <1>      db  000h, 000h, 07ch, 0c6h, 0fch, 0c6h, 0c6h, 0fch, 0c0h, 0c0h, 040h, 000h, 000h, 000h, 0feh, 0c6h
448 00014111 C040000000FEC6      <1>
449 00014118 C6C0C0C0C0C0C00000- <1>      db  0c6h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 06ch
449 00014121 0000000000FE6C      <1>
450 00014128 6C6C6C6C6C00000000- <1>      db  06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 0feh, 0c6h, 060h, 030h, 018h, 030h
450 00014131 00FEC660301830      <1>
451 00014138 60C6FE000000000000- <1>      db  060h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 0d8h
451 00014141 00007ED8D8D8D8      <1>
452 00014148 700000000000000066- <1>      db  070h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0c0h
452 00014151 6666667C6060C0      <1>
453 00014158 00000000000076DC18- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h
453 00014161 18181818000000      <1>
454 00014168 00007E183C6666663C- <1>      db  000h, 000h, 07eh, 018h, 03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h, 000h
454 00014171 187E0000000000      <1>
455 00014178 386CC6C6FEC6C66C38- <1>      db  038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 038h, 06ch
455 00014181 0000000000386C      <1>
456 00014188 C6C6C66C6C6CEE0000- <1>      db  0c6h, 0c6h, 0c6h, 06ch, 06ch, 06ch, 0eeh, 000h, 000h, 000h, 000h, 000h, 01eh, 030h, 018h, 00ch
456 00014191 0000001E30180C      <1>
457 00014198 3E6666663C00000000- <1>      db  03eh, 066h, 066h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 0dbh, 0dbh
457 000141A1 000000007EDBDB      <1>
458 000141A8 7E0000000000000003- <1>      db  07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 003h, 006h, 07eh, 0dbh, 0dbh, 0f3h, 07eh, 060h
458 000141B1 067EDBDBF37E60      <1>
459 000141B8 C000000000001C3060- <1>      db  0c0h, 000h, 000h, 000h, 000h, 000h, 01ch, 030h, 060h, 060h, 07ch, 060h, 060h, 030h, 01ch, 000h
459 000141C1 607C6060301C00      <1>
460 000141C8 00000000007CC6C6C6- <1>      db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h
460 000141D1 C6C6C6C6000000      <1>
461 000141D8 000000FE0000FE0000- <1>      db  000h, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h
461 000141E1 FE000000000000      <1>
462 000141E8 0018187E18180000FF- <1>      db  000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 030h, 018h
462 000141F1 00000000003018      <1>
463 000141F8 0C060C1830007E0000- <1>      db  00ch, 006h, 00ch, 018h, 030h, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 00ch, 018h, 030h, 060h
463 00014201 0000000C183060      <1>
464 00014208 30180C007E00000000- <1>      db  030h, 018h, 00ch, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 00eh, 01bh, 01bh, 018h, 018h, 018h
464 00014211 000E1B1B181818      <1>
465 00014218 181818181818181818- <1>      db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h
465 00014221 1818181818D8D8      <1>
466 00014228 700000000000001818- <1>      db  070h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 07eh, 000h, 018h, 018h, 000h, 000h
466 00014231 007E0018180000      <1>
467 00014238 00000000000076DC00- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h
467 00014241 76DC0000000000      <1>
468 00014248 00386C6C3800000000- <1>      db  000h, 038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
468 00014251 00000000000000      <1>
469 00014258 000000001818000000- <1>      db  000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
469 00014261 00000000000000      <1>
470 00014268 000000180000000000- <1>      db  000h, 000h, 000h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 00fh, 00ch, 00ch, 00ch, 00ch
470 00014271 00000F0C0C0C0C      <1>
471 00014278 0CEC6C3C1C00000000- <1>      db  00ch, 0ech, 06ch, 03ch, 01ch, 000h, 000h, 000h, 000h, 0d8h, 06ch, 06ch, 06ch, 06ch, 06ch, 000h
471 00014281 D86C6C6C6C6C00      <1>
472 00014288 0000000000000070D8- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 070h, 0d8h, 030h, 060h, 0c8h, 0f8h, 000h, 000h, 000h
472 00014291 3060C8F8000000      <1>
473 00014298 00000000000000007C- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 000h, 000h
```

```
473 000142A1 7C7C7C7C7C0000      <1>
474 000142A8 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
474 000142B1 00000000000000      <1>
475                              <1> vgafont16:
476 000142B8 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
476 000142C1 00000000000000      <1>
477 000142C8 00007E81A58181BD99- <1>     db  000h, 000h, 07eh, 081h, 0a5h, 081h, 081h, 0bdh, 099h, 081h, 081h, 07eh, 000h, 000h, 000h, 000h
477 000142D1 81817E00000000      <1>
478 000142D8 00007EFFDBFFFFC3E7- <1>     db  000h, 000h, 07eh, 0ffh, 0dbh, 0ffh, 0ffh, 0c3h, 0e7h, 0ffh, 0ffh, 07eh, 000h, 000h, 000h, 000h
478 000142E1 FFFF7E00000000      <1>
479 000142E8 000000006CFEFEFEFE- <1>     db  000h, 000h, 000h, 000h, 06ch, 0feh, 0feh, 0feh, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h
479 000142F1 7C381000000000      <1>
480 000142F8 0000000010387CFE7C- <1>     db  000h, 000h, 000h, 000h, 010h, 038h, 07ch, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h, 000h
480 00014301 38100000000000      <1>
481 00014308 000000183C3CE7E7E7- <1>     db  000h, 000h, 000h, 018h, 03ch, 03ch, 0e7h, 0e7h, 0e7h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
481 00014311 18183C00000000      <1>
482 00014318 000000183C7EFFFF7E- <1>     db  000h, 000h, 000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 07eh, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
482 00014321 18183C00000000      <1>
483 00014328 000000000000183C3C- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h
483 00014331 18000000000000      <1>
484 00014338 FFFFFFFFFFFFE7C3C3- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
484 00014341 E7FFFFFFFFFFFF      <1>
485 00014348 00000000003C664242- <1>     db  000h, 000h, 000h, 000h, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h
485 00014351 663C0000000000      <1>
486 00014358 FFFFFFFFFFC399BDBD- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
486 00014361 99C3FFFFFFFFFF      <1>
487 00014368 00001E0E1A3278CCCC- <1>     db  000h, 000h, 01eh, 00eh, 01ah, 032h, 078h, 0cch, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h
487 00014371 CCCC7800000000      <1>
488 00014378 00003C666666663C18- <1>     db  000h, 000h, 03ch, 066h, 066h, 066h, 066h, 03ch, 018h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h
488 00014381 7E181800000000      <1>
489 00014388 00003F333F30303030- <1>     db  000h, 000h, 03fh, 033h, 03fh, 030h, 030h, 030h, 030h, 070h, 0f0h, 0e0h, 000h, 000h, 000h, 000h
489 00014391 70F0E000000000      <1>
490 00014398 00007F637F63636363- <1>     db  000h, 000h, 07fh, 063h, 07fh, 063h, 063h, 063h, 063h, 067h, 0e7h, 0e6h, 0c0h, 000h, 000h, 000h
490 000143A1 67E7E6C0000000      <1>
491 000143A8 0000001818DB3CE73C- <1>     db  000h, 000h, 000h, 018h, 018h, 0dbh, 03ch, 0e7h, 03ch, 0dbh, 018h, 018h, 000h, 000h, 000h, 000h
491 000143B1 DB181800000000      <1>
492 000143B8 0080C0E0F0F8FEF8F0- <1>     db  000h, 080h, 0c0h, 0e0h, 0f0h, 0f8h, 0feh, 0f8h, 0f0h, 0e0h, 0c0h, 080h, 000h, 000h, 000h, 000h
492 000143C1 E0C08000000000      <1>
493 000143C8 0002060E1E3EFE3E1E- <1>     db  000h, 002h, 006h, 00eh, 01eh, 03eh, 0feh, 03eh, 01eh, 00eh, 006h, 002h, 000h, 000h, 000h, 000h
493 000143D1 0E060200000000      <1>
494 000143D8 0000183C7E1818187E- <1>     db  000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h
494 000143E1 3C180000000000      <1>
495 000143E8 000066666666666666- <1>     db  000h, 000h, 066h, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 066h, 066h, 000h, 000h, 000h, 000h
495 000143F1 00666600000000      <1>
496 000143F8 00007FDBDBDB7B1B1B- <1>     db  000h, 000h, 07fh, 0dbh, 0dbh, 0dbh, 07bh, 01bh, 01bh, 01bh, 01bh, 01bh, 000h, 000h, 000h, 000h
496 00014401 1B1B1B00000000      <1>
497 00014408 007CC660386CC6C66C- <1>     db  000h, 07ch, 0c6h, 060h, 038h, 06ch, 0c6h, 0c6h, 06ch, 038h, 00ch, 0c6h, 07ch, 000h, 000h, 000h
497 00014411 380CC67C000000      <1>
498 00014418 0000000000000000FE- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 0feh, 0feh, 000h, 000h, 000h, 000h
498 00014421 FEFEFE00000000      <1>
499 00014428 0000183C7E1818187E- <1>     db  000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h
499 00014431 3C187E00000000      <1>
500 00014438 0000183C7E18181818- <1>     db  000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
500 00014441 18181800000000      <1>
501 00014448 000018181818181818- <1>     db  000h, 000h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h
501 00014451 7E3C1800000000      <1>
502 00014458 0000000000180CFE0C- <1>     db  000h, 000h, 000h, 000h, 000h, 018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h
502 00014461 18000000000000      <1>
503 00014468 00000000003060FE60- <1>     db  000h, 000h, 000h, 000h, 000h, 030h, 060h, 0feh, 060h, 030h, 000h, 000h, 000h, 000h, 000h, 000h
503 00014471 30000000000000      <1>
504 00014478 000000000000C0C0C0- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c0h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h
504 00014481 FE000000000000      <1>
505 00014488 00000000002466FF66- <1>     db  000h, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 000h
505 00014491 24000000000000      <1>
506 00014498 000000001038387C7C- <1>     db  000h, 000h, 000h, 000h, 010h, 038h, 038h, 07ch, 07ch, 0feh, 0feh, 000h, 000h, 000h, 000h, 000h
506 000144A1 FEFE0000000000      <1>
507 000144A8 00000000FEFE7C7C38- <1>     db  000h, 000h, 000h, 000h, 0feh, 0feh, 07ch, 07ch, 038h, 038h, 010h, 000h, 000h, 000h, 000h, 000h
507 000144B1 38100000000000      <1>
508 000144B8 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
508 000144C1 00000000000000      <1>
509 000144C8 0000183C3C3C181818- <1>     db  000h, 000h, 018h, 03ch, 03ch, 03ch, 018h, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h
509 000144D1 00181800000000      <1>
510 000144D8 006666662400000000- <1>     db  000h, 066h, 066h, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
510 000144E1 00000000000000      <1>
511 000144E8 0000006C6CFE6C6C6C- <1>     db  000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch, 06ch, 06ch, 0feh, 06ch, 06ch, 000h, 000h, 000h, 000h
511 000144F1 FE6C6C00000000      <1>
512 000144F8 18187CC6C2C07C0606- <1>     db  018h, 018h, 07ch, 0c6h, 0c2h, 0c0h, 07ch, 006h, 006h, 086h, 0c6h, 07ch, 018h, 018h, 000h, 000h
512 00014501 86C67C18180000      <1>
513 00014508 00000000C2C60C1830- <1>     db  000h, 000h, 000h, 000h, 0c2h, 0c6h, 00ch, 018h, 030h, 060h, 0c6h, 086h, 000h, 000h, 000h, 000h
513 00014511 60C68600000000      <1>
514 00014518 0000386C6C3876DCCC- <1>     db  000h, 000h, 038h, 06ch, 06ch, 038h, 076h, 0dch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
514 00014521 CCCC7600000000      <1>
515 00014528 003030306000000000- <1>     db  000h, 030h, 030h, 030h, 060h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
515 00014531 00000000000000      <1>
516 00014538 00000C183030303030- <1>     db  000h, 000h, 00ch, 018h, 030h, 030h, 030h, 030h, 030h, 030h, 018h, 00ch, 000h, 000h, 000h, 000h
516 00014541 30180C00000000      <1>
517 00014548 000030180C0C0C0C0C- <1>     db  000h, 000h, 030h, 018h, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 018h, 030h, 000h, 000h, 000h, 000h
517 00014551 0C183000000000      <1>
518 00014558 0000000000663CFF3C- <1>     db  000h, 000h, 000h, 000h, 000h, 066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 000h, 000h, 000h
518 00014561 66000000000000      <1>
519 00014568 000000000018187E18- <1>     db  000h, 000h, 000h, 000h, 000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h
519 00014571 18000000000000      <1>
520 00014578 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 030h, 000h, 000h, 000h
520 00014581 18181830000000      <1>
521 00014588 00000000000000FE00- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
521 00014591 00000000000000      <1>
522 00014598 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h
522 000145A1 00181800000000      <1>
523 000145A8 0000000002060C1830- <1>     db  000h, 000h, 000h, 000h, 002h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h, 000h, 000h, 000h
523 000145B1 60C08000000000      <1>
524 000145B8 00003C66C3C3DBDBC3- <1>     db  000h, 000h, 03ch, 066h, 0c3h, 0c3h, 0dbh, 0dbh, 0c3h, 0c3h, 066h, 03ch, 000h, 000h, 000h, 000h
524 000145C1 C3663C00000000      <1>
525 000145C8 000018387818181818- <1>     db  000h, 000h, 018h, 038h, 078h, 018h, 018h, 018h, 018h, 018h, 018h, 07eh, 000h, 000h, 000h, 000h
525 000145D1 18187E00000000      <1>
526 000145D8 00007CC6060C183060- <1>     db  000h, 000h, 07ch, 0c6h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 0c6h, 0feh, 000h, 000h, 000h, 000h
526 000145E1 C0C6FE00000000      <1>
527 000145E8 00007CC606063C0606- <1>     db  000h, 000h, 07ch, 0c6h, 006h, 006h, 03ch, 006h, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h
527 000145F1 06C67C00000000      <1>
528 000145F8 00000C1C3C6CCCFE0C- <1>     db  000h, 000h, 00ch, 01ch, 03ch, 06ch, 0cch, 0feh, 00ch, 00ch, 00ch, 01eh, 000h, 000h, 000h, 000h
```

```
528 00014601 0C0C1E00000000      <1>
529 00014608 0000FEC0C0C0FC0606- <1>      db  000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0fch, 006h, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h
529 00014611 06C67C00000000      <1>
530 00014618 00003860C0C0FCC6C6- <1>      db  000h, 000h, 038h, 060h, 0c0h, 0c0h, 0fch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
530 00014621 C6C67C00000000      <1>
531 00014628 0000FEC606060C1830- <1>      db  000h, 000h, 0feh, 0c6h, 006h, 006h, 00ch, 018h, 030h, 030h, 030h, 030h, 000h, 000h, 000h, 000h
531 00014631 30303000000000      <1>
532 00014638 00007CC6C6C67CC6C6- <1>      db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
532 00014641 C6C67C00000000      <1>
533 00014648 00007CC6C6C67E0606- <1>      db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07eh, 006h, 006h, 006h, 00ch, 078h, 000h, 000h, 000h, 000h
533 00014651 060C7800000000      <1>
534 00014658 000000001818000000- <1>      db  000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h
534 00014661 18180000000000      <1>
535 00014668 000000001818000000- <1>      db  000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 018h, 018h, 030h, 000h, 000h, 000h, 000h
535 00014671 18183000000000      <1>
536 00014678 000000060C18306030- <1>      db  000h, 000h, 000h, 006h, 00ch, 018h, 030h, 060h, 030h, 018h, 00ch, 006h, 000h, 000h, 000h, 000h
536 00014681 180C0600000000      <1>
537 00014688 00000000007E00007E- <1>      db  000h, 000h, 000h, 000h, 000h, 07eh, 000h, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h
537 00014691 00000000000000      <1>
538 00014698 0000006030180C060C- <1>      db  000h, 000h, 000h, 060h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 060h, 000h, 000h, 000h, 000h
538 000146A1 18306000000000      <1>
539 000146A8 00007CC6C60C181818- <1>      db  000h, 000h, 07ch, 0c6h, 0c6h, 00ch, 018h, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h
539 000146B1 00181800000000      <1>
540 000146B8 0000007CC6C6DEDEDE- <1>      db  000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0deh, 0deh, 0deh, 0dch, 0c0h, 07ch, 000h, 000h, 000h, 000h
540 000146C1 DCC07C00000000      <1>
541 000146C8 000010386CC6C6FEC6- <1>      db  000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
541 000146D1 C6C6C600000000      <1>
542 000146D8 0000FC6666667C6666- <1>      db  000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 066h, 066h, 066h, 066h, 0fch, 000h, 000h, 000h, 000h
542 000146E1 6666FC00000000      <1>
543 000146E8 00003C66C2C0C0C0C0- <1>      db  000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 000h, 000h, 000h, 000h
543 000146F1 C2663C00000000      <1>
544 000146F8 0000F86C6666666666- <1>      db  000h, 000h, 0f8h, 06ch, 066h, 066h, 066h, 066h, 066h, 066h, 06ch, 0f8h, 000h, 000h, 000h, 000h
544 00014701 666CF800000000      <1>
545 00014708 0000FE666268786860- <1>      db  000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 062h, 066h, 0feh, 000h, 000h, 000h, 000h
545 00014711 6266FE00000000      <1>
546 00014718 0000FE666268786860- <1>      db  000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
546 00014721 6060F000000000      <1>
547 00014728 00003C66C2C0C0DEC6- <1>      db  000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0deh, 0c6h, 0c6h, 066h, 03ah, 000h, 000h, 000h, 000h
547 00014731 C6663A00000000      <1>
548 00014738 0000C6C6C6C6FEC6C6- <1>      db  000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
548 00014741 C6C6C600000000      <1>
549 00014748 00003C181818181818- <1>      db  000h, 000h, 03ch, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
549 00014751 18183C00000000      <1>
550 00014758 00001E0C0C0C0C0CCC- <1>      db  000h, 000h, 01eh, 00ch, 00ch, 00ch, 00ch, 00ch, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h
550 00014761 CCCC7800000000      <1>
551 00014768 0000E666666C78786C- <1>      db  000h, 000h, 0e6h, 066h, 066h, 06ch, 078h, 078h, 06ch, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h
551 00014771 6666E600000000      <1>
552 00014778 0000F0606060606060- <1>      db  000h, 000h, 0f0h, 060h, 060h, 060h, 060h, 060h, 060h, 062h, 066h, 0feh, 000h, 000h, 000h, 000h
552 00014781 6266FE00000000      <1>
553 00014788 0000C3E7FFFFDBC3C3- <1>      db  000h, 000h, 0c3h, 0e7h, 0ffh, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h, 000h, 000h, 000h
553 00014791 C3C3C300000000      <1>
554 00014798 0000C6E6F6FEDECEC6- <1>      db  000h, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
554 000147A1 C6C6C600000000      <1>
555 000147A8 00007CC6C6C6C6C6C6- <1>      db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
555 000147B1 C6C67C00000000      <1>
556 000147B8 0000FC6666667C6060- <1>      db  000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 060h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
556 000147C1 6060F000000000      <1>
557 000147C8 00007CC6C6C6C6C6C6- <1>      db  000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0deh, 07ch, 00ch, 00eh, 000h, 000h
557 000147D1 D6DE7C0C0E0000      <1>
558 000147D8 0000FC6666667C6C66- <1>      db  000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 06ch, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h
558 000147E1 6666E600000000      <1>
559 000147E8 00007CC6C660380C06- <1>      db  000h, 000h, 07ch, 0c6h, 0c6h, 060h, 038h, 00ch, 006h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
559 000147F1 C6C67C00000000      <1>
560 000147F8 0000FFDB9918181818- <1>      db  000h, 000h, 0ffh, 0dbh, 099h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
560 00014801 18183C00000000      <1>
561 00014808 0000C6C6C6C6C6C6C6- <1>      db  000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
561 00014811 C6C67C00000000      <1>
562 00014818 0000C3C3C3C3C3C3C3- <1>      db  000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h
562 00014821 663C1800000000      <1>
563 00014828 0000C3C3C3C3C3DBDB- <1>      db  000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 066h, 000h, 000h, 000h, 000h
563 00014831 FF666600000000      <1>
564 00014838 0000C3C3663C18183C- <1>      db  000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 03ch, 066h, 0c3h, 0c3h, 000h, 000h, 000h, 000h
564 00014841 66C3C300000000      <1>
565 00014848 0000C3C3C3663C1818- <1>      db  000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
565 00014851 18183C00000000      <1>
566 00014858 0000FFC3860C183060- <1>      db  000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h, 030h, 060h, 0c1h, 0c3h, 0ffh, 000h, 000h, 000h, 000h
566 00014861 C1C3FF00000000      <1>
567 00014868 00003C303030303030- <1>      db  000h, 000h, 03ch, 030h, 030h, 030h, 030h, 030h, 030h, 030h, 030h, 03ch, 000h, 000h, 000h, 000h
567 00014871 30303C00000000      <1>
568 00014878 00000080C0E070381C- <1>      db  000h, 000h, 000h, 080h, 0c0h, 0e0h, 070h, 038h, 01ch, 00eh, 006h, 002h, 000h, 000h, 000h, 000h
568 00014881 0E060200000000      <1>
569 00014888 00003C0C0C0C0C0C0C- <1>      db  000h, 000h, 03ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 03ch, 000h, 000h, 000h, 000h
569 00014891 0C0C3C00000000      <1>
570 00014898 10386CC6000000000- <1>      db  010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
570 000148A1 00000000000000      <1>
571 000148A8 000000000000000000- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 000h
571 000148B1 00000000FF0000      <1>
572 000148B8 303018000000000000- <1>      db  030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
572 000148C1 00000000000000      <1>
573 000148C8 0000000000780C7CCC- <1>      db  000h, 000h, 000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
573 000148D1 CCCC7600000000      <1>
574 000148D8 0000E06060786C6666- <1>      db  000h, 000h, 0e0h, 060h, 060h, 078h, 06ch, 066h, 066h, 066h, 066h, 07ch, 000h, 000h, 000h, 000h
574 000148E1 66667C00000000      <1>
575 000148E8 00000000007CC6C0C0- <1>      db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c0h, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
575 000148F1 C0C67C00000000      <1>
576 000148F8 00001C0C0C3C6CCCCC- <1>      db  000h, 000h, 01ch, 00ch, 00ch, 03ch, 06ch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
576 00014901 CCCC7600000000      <1>
577 00014908 00000000007CC6FEC0- <1>      db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
577 00014911 C0C67C00000000      <1>
578 00014918 0000386C6460F06060- <1>      db  000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
578 00014921 6060F000000000      <1>
579 00014928 000000000076CCCCCC- <1>      db  000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 0cch, 0cch, 07ch, 00ch, 0cch, 078h, 000h
579 00014931 CCCC7C0CCC7800      <1>
580 00014938 0000E060606C766666- <1>      db  000h, 000h, 0e0h, 060h, 060h, 06ch, 076h, 066h, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h
580 00014941 6666E600000000      <1>
581 00014948 000018180038181818- <1>      db  000h, 000h, 018h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
581 00014951 18183C00000000      <1>
582 00014958 00000606000E060606- <1>      db  000h, 000h, 006h, 006h, 000h, 00eh, 006h, 006h, 006h, 006h, 006h, 006h, 066h, 066h, 03ch, 000h
582 00014961 06060666663C00      <1>
```

```
583 00014968 0000E06060666C7878- <1>     db  000h, 000h, 0e0h, 060h, 060h, 066h, 06ch, 078h, 078h, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h
583 00014971 6C66E600000000      <1>
584 00014978 000038181818181818- <1>     db  000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
584 00014981 18183C00000000      <1>
585 00014988 0000000000E6FFDBDB- <1>     db  000h, 000h, 000h, 000h, 000h, 0e6h, 0ffh, 0dbh, 0dbh, 0dbh, 0dbh, 0dbh, 000h, 000h, 000h, 000h
585 00014991 DBDBDB00000000      <1>
586 00014998 0000000000DC666666- <1>     db  000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 000h, 000h, 000h
586 000149A1 66666600000000      <1>
587 000149A8 00000000007CC6C6C6- <1>     db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
587 000149B1 C6C67C00000000      <1>
588 000149B8 0000000000DC666666- <1>     db  000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h
588 000149C1 66667C6060F000      <1>
589 000149C8 000000000076CCCCCC- <1>     db  000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 0cch, 0cch, 07ch, 00ch, 00ch, 01eh, 000h
589 000149D1 CCCC7C0C0C1E00      <1>
590 000149D8 0000000000DC766660- <1>     db  000h, 000h, 000h, 000h, 000h, 0dch, 076h, 066h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
590 000149E1 6060F000000000      <1>
591 000149E8 00000000007CC66038- <1>     db  000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 060h, 038h, 00ch, 0c6h, 07ch, 000h, 000h, 000h, 000h
591 000149F1 0CC67C00000000      <1>
592 000149F8 0000103030FC303030- <1>     db  000h, 000h, 010h, 030h, 030h, 0fch, 030h, 030h, 030h, 030h, 036h, 01ch, 000h, 000h, 000h, 000h
592 00014A01 30361C00000000      <1>
593 00014A08 0000000000CCCCCCCC- <1>     db  000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
593 00014A11 CCCC7600000000      <1>
594 00014A18 0000000000C3C3C3C3- <1>     db  000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h
594 00014A21 663C1800000000      <1>
595 00014A28 0000000000C3C3C3DB- <1>     db  000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h, 000h, 000h, 000h
595 00014A31 DBFF6600000000      <1>
596 00014A38 0000000000C3663C18- <1>     db  000h, 000h, 000h, 000h, 000h, 0c3h, 066h, 03ch, 018h, 03ch, 066h, 0c3h, 000h, 000h, 000h, 000h
596 00014A41 3C66C300000000      <1>
597 00014A48 0000000000C6C6C6C6- <1>     db  000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 0f8h, 000h
597 00014A51 C6C67E060CF800      <1>
598 00014A58 0000000000FECC1830- <1>     db  000h, 000h, 000h, 000h, 000h, 0feh, 0cch, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h
598 00014A61 60C6FE00000000      <1>
599 00014A68 00000E181818701818- <1>     db  000h, 000h, 00eh, 018h, 018h, 018h, 070h, 018h, 018h, 018h, 018h, 00eh, 000h, 000h, 000h, 000h
599 00014A71 18180E00000000      <1>
600 00014A78 000018181818001818- <1>     db  000h, 000h, 018h, 018h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
600 00014A81 18181800000000      <1>
601 00014A88 0000701818180E1818- <1>     db  000h, 000h, 070h, 018h, 018h, 018h, 00eh, 018h, 018h, 018h, 018h, 070h, 000h, 000h, 000h, 000h
601 00014A91 18187000000000      <1>
602 00014A98 000076DC0000000000- <1>     db  000h, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
602 00014AA1 00000000000000      <1>
603 00014AA8 0000000010386CC6C6- <1>     db  000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h
603 00014AB1 C6FE0000000000      <1>
604 00014AB8 00003C66C2C0C0C0C2- <1>     db  000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 00ch, 006h, 07ch, 000h, 000h
604 00014AC1 663C0C067C0000      <1>
605 00014AC8 0000CC0000CCCCCCCC- <1>     db  000h, 000h, 0cch, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
605 00014AD1 CCCC7600000000      <1>
606 00014AD8 000C1830007CC6FEC0- <1>     db  000h, 00ch, 018h, 030h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
606 00014AE1 C0C67C00000000      <1>
607 00014AE8 0010386C00780C7CCC- <1>     db  000h, 010h, 038h, 06ch, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
607 00014AF1 CCCC7600000000      <1>
608 00014AF8 0000CC0000780C7CCC- <1>     db  000h, 000h, 0cch, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
608 00014B01 CCCC7600000000      <1>
609 00014B08 0060301800780C7CCC- <1>     db  000h, 060h, 030h, 018h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
609 00014B11 CCCC7600000000      <1>
610 00014B18 00386C3800780C7CCC- <1>     db  000h, 038h, 06ch, 038h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
610 00014B21 CCCC7600000000      <1>
611 00014B28 000000003C66606066- <1>     db  000h, 000h, 000h, 000h, 03ch, 066h, 060h, 060h, 066h, 03ch, 00ch, 006h, 03ch, 000h, 000h, 000h
611 00014B31 3C0C063C000000      <1>
612 00014B38 0010386C007CC6FEC0- <1>     db  000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
612 00014B41 C0C67C00000000      <1>
613 00014B48 0000C600007CC6FEC0- <1>     db  000h, 000h, 0c6h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
613 00014B51 C0C67C00000000      <1>
614 00014B58 00603018007CC6FEC0- <1>     db  000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
614 00014B61 C0C67C00000000      <1>
615 00014B68 000066000038181818- <1>     db  000h, 000h, 066h, 000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
615 00014B71 18183C00000000      <1>
616 00014B78 00183C660038181818- <1>     db  000h, 018h, 03ch, 066h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
616 00014B81 18183C00000000      <1>
617 00014B88 006030180038181818- <1>     db  000h, 060h, 030h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
617 00014B91 18183C00000000      <1>
618 00014B98 00C60010386CC6C6FE- <1>     db  000h, 0c6h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
618 00014BA1 C6C6C600000000      <1>
619 00014BA8 386C3800386CC6C6FE- <1>     db  038h, 06ch, 038h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
619 00014BB1 C6C6C600000000      <1>
620 00014BB8 18306000FE66607C60- <1>     db  018h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 060h, 066h, 0feh, 000h, 000h, 000h, 000h
620 00014BC1 6066FE00000000      <1>
621 00014BC8 00000000006E3B1B7E- <1>     db  000h, 000h, 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h, 000h, 000h
621 00014BD1 D8DC7700000000      <1>
622 00014BD8 00003E6CCCCCFECCCC- <1>     db  000h, 000h, 03eh, 06ch, 0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h
622 00014BE1 CCCCCE00000000      <1>
623 00014BE8 0010386C007CC6C6C6- <1>     db  000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
623 00014BF1 C6C67C00000000      <1>
624 00014BF8 0000C600007CC6C6C6- <1>     db  000h, 000h, 0c6h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
624 00014C01 C6C67C00000000      <1>
625 00014C08 00603018007CC6C6C6- <1>     db  000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
625 00014C11 C6C67C00000000      <1>
626 00014C18 003078CC00CCCCCCCC- <1>     db  000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
626 00014C21 CCCC7600000000      <1>
627 00014C28 0060301800CCCCCCCC- <1>     db  000h, 060h, 030h, 018h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
627 00014C31 CCCC7600000000      <1>
628 00014C38 0000C60000C6C6C6C6- <1>     db  000h, 000h, 0c6h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h
628 00014C41 C6C67E060C7800      <1>
629 00014C48 00C6007CC6C6C6C6C6- <1>     db  000h, 0c6h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
629 00014C51 C6C67C00000000      <1>
630 00014C58 00C600C6C6C6C6C6C6- <1>     db  000h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
630 00014C61 C6C67C00000000      <1>
631 00014C68 0018187EC3C0C0C0C3- <1>     db  000h, 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h
631 00014C71 7E181800000000      <1>
632 00014C78 00386C6460F0606060- <1>     db  000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 060h, 0e6h, 0fch, 000h, 000h, 000h, 000h
632 00014C81 60E6FC00000000      <1>
633 00014C88 0000C3663C18FF18FF- <1>     db  000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h
633 00014C91 18181800000000      <1>
634 00014C98 00FC66667C62666F66- <1>     db  000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 066h, 0f3h, 000h, 000h, 000h, 000h
634 00014CA1 6666F300000000      <1>
635 00014CA8 000E1B1818187E1818- <1>     db  000h, 00eh, 01bh, 018h, 018h, 018h, 07eh, 018h, 018h, 018h, 018h, 018h, 0d8h, 070h, 000h, 000h
635 00014CB1 181818D8700000      <1>
636 00014CB8 0018306000780C7CCC- <1>     db  000h, 018h, 030h, 060h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
636 00014CC1 CCCC7600000000      <1>
637 00014CC8 000C18300038181818- <1>     db  000h, 00ch, 018h, 030h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
```

```
637 00014CD1 18183C00000000       <1>
638 00014CD8 00183060007CC6C6C6- <1>      db  000h, 018h, 030h, 060h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
638 00014CE1 C6C67C00000000       <1>
639 00014CE8 0018306000CCCCCCCC- <1>      db  000h, 018h, 030h, 060h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
639 00014CF1 CCCC7600000000       <1>
640 00014CF8 000076DC00DC666666- <1>      db  000h, 000h, 076h, 0dch, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 000h, 000h, 000h
640 00014D01 66666600000000       <1>
641 00014D08 76DC00C6E6F6FEDECE- <1>      db  076h, 0dch, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
641 00014D11 C6C6C600000000       <1>
642 00014D18 003C6C6C3E007E0000- <1>      db  000h, 03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
642 00014D21 00000000000000       <1>
643 00014D28 00386C6C38007C0000- <1>      db  000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
643 00014D31 00000000000000       <1>
644 00014D38 0000303000303060C0- <1>      db  000h, 000h, 030h, 030h, 000h, 030h, 030h, 060h, 0c0h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
644 00014D41 C6C67C00000000       <1>
645 00014D48 000000000000FEC0C0- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h
645 00014D51 C0C00000000000       <1>
646 00014D58 000000000000FE0606- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 0feh, 006h, 006h, 006h, 006h, 000h, 000h, 000h, 000h, 000h
646 00014D61 06060000000000       <1>
647 00014D68 00C0C0C2C6CC183060- <1>      db  000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 060h, 0ceh, 09bh, 006h, 00ch, 01fh, 000h, 000h
647 00014D71 CE9B060C1F0000       <1>
648 00014D78 00C0C0C2C6CC183066- <1>      db  000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 066h, 0ceh, 096h, 03eh, 006h, 006h, 000h, 000h
648 00014D81 CE963E06060000       <1>
649 00014D88 00001818001818183C- <1>      db  000h, 000h, 018h, 018h, 000h, 018h, 018h, 018h, 03ch, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h
649 00014D91 3C3C1800000000       <1>
650 00014D98 0000000000366CD86C- <1>      db  000h, 000h, 000h, 000h, 000h, 036h, 06ch, 0d8h, 06ch, 036h, 000h, 000h, 000h, 000h, 000h, 000h
650 00014DA1 36000000000000       <1>
651 00014DA8 0000000000D86C366C- <1>      db  000h, 000h, 000h, 000h, 000h, 0d8h, 06ch, 036h, 06ch, 0d8h, 000h, 000h, 000h, 000h, 000h, 000h
651 00014DB1 D8000000000000       <1>
652 00014DB8 114411441144114411- <1>      db  011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h
652 00014DC1 44114411441144       <1>
653 00014DC8 55AA55AA55AA55AA55- <1>      db  055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah
653 00014DD1 AA55AA55AA55AA       <1>
654 00014DD8 DD77DD77DD77DD77DD- <1>      db  0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h
654 00014DE1 77DD77DD77DD77       <1>
655 00014DE8 181818181818181818- <1>      db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
655 00014DF1 18181818181818       <1>
656 00014DF8 18181818181818F818- <1>      db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
656 00014E01 18181818181818       <1>
657 00014E08 1818181818F818F818- <1>      db  018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
657 00014E11 18181818181818       <1>
658 00014E18 36363636363636F636- <1>      db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
658 00014E21 36363636363636       <1>
659 00014E28 00000000000000FE36- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
659 00014E31 36363636363636       <1>
660 00014E38 0000000000F818F818- <1>      db  000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
660 00014E41 18181818181818       <1>
661 00014E48 3636363636F606F636- <1>      db  036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
661 00014E51 36363636363636       <1>
662 00014E58 363636363636363636- <1>      db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
662 00014E61 36363636363636       <1>
663 00014E68 0000000000FE06F636- <1>      db  000h, 000h, 000h, 000h, 000h, 0feh, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
663 00014E71 36363636363636       <1>
664 00014E78 3636363636F606FE00- <1>      db  036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
664 00014E81 00000000000000       <1>
665 00014E88 36363636363636FE00- <1>      db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
665 00014E91 00000000000000       <1>
666 00014E98 1818181818F818F800- <1>      db  018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
666 00014EA1 00000000000000       <1>
667 00014EA8 00000000000000F818- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
667 00014EB1 18181818181818       <1>
668 00014EB8 181818181818181F00- <1>      db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
668 00014EC1 00000000000000       <1>
669 00014EC8 18181818181818FF00- <1>      db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
669 00014ED1 00000000000000       <1>
670 00014ED8 00000000000000FF18- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
670 00014EE1 18181818181818       <1>
671 00014EE8 181818181818181F18- <1>      db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
671 00014EF1 18181818181818       <1>
672 00014EF8 00000000000000FF00- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
672 00014F01 00000000000000       <1>
673 00014F08 18181818181818FF18- <1>      db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
673 00014F11 18181818181818       <1>
674 00014F18 18181818181F181F18- <1>      db  018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
674 00014F21 18181818181818       <1>
675 00014F28 363636363636363736- <1>      db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
675 00014F31 36363636363636       <1>
676 00014F38 363636363637303F00- <1>      db  036h, 036h, 036h, 036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
676 00014F41 00000000000000       <1>
677 00014F48 00000000003F303736- <1>      db  000h, 000h, 000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
677 00014F51 36363636363636       <1>
678 00014F58 3636363636F700FF00- <1>      db  036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
678 00014F61 00000000000000       <1>
679 00014F68 0000000000FF00F736- <1>      db  000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
679 00014F71 36363636363636       <1>
680 00014F78 363636363637303736- <1>      db  036h, 036h, 036h, 036h, 036h, 037h, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
680 00014F81 36363636363636       <1>
681 00014F88 0000000000FF00FF00- <1>      db  000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
681 00014F91 00000000000000       <1>
682 00014F98 3636363636F700F736- <1>      db  036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
682 00014FA1 36363636363636       <1>
683 00014FA8 1818181818FF00FF00- <1>      db  018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
683 00014FB1 00000000000000       <1>
684 00014FB8 36363636363636FF00- <1>      db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
684 00014FC1 00000000000000       <1>
685 00014FC8 0000000000FF00FF18- <1>      db  000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
685 00014FD1 18181818181818       <1>
686 00014FD8 00000000000000FF36- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
686 00014FE1 36363636363636       <1>
687 00014FE8 363636363636363F00- <1>      db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
687 00014FF1 00000000000000       <1>
688 00014FF8 18181818181F181F00- <1>      db  018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
688 00015001 00000000000000       <1>
689 00015008 00000000001F181F18- <1>      db  000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
689 00015011 18181818181818       <1>
690 00015018 000000000000003F36- <1>      db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
690 00015021 36363636363636       <1>
691 00015028 36363636363636FF36- <1>      db  036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
691 00015031 36363636363636       <1>
```

```
692 00015038 1818181818FF18FF18- <1>     db  018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
692 00015041 18181818181818      <1>
693 00015048 18181818181818F800- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
693 00015051 00000000000000      <1>
694 00015058 000000000000001F18- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
694 00015061 18181818181818      <1>
695 00015068 FFFFFFFFFFFFFFFFFF- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
695 00015071 FFFFFFFFFFFFFF      <1>
696 00015078 00000000000000FFFF- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
696 00015081 FFFFFFFFFFFFFF      <1>
697 00015088 F0F0F0F0F0F0F0F0F0- <1>     db  0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h
697 00015091 F0F0F0F0F0F0F0      <1>
698 00015098 0F0F0F0F0F0F0F0F0F- <1>     db  00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh
698 000150A1 0F0F0F0F0F0F0F      <1>
699 000150A8 FFFFFFFFFFFFFF0000- <1>     db  0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
699 000150B1 00000000000000      <1>
700 000150B8 000000000076DCD8D8- <1>     db  000h, 000h, 000h, 000h, 000h, 076h, 0dch, 0d8h, 0d8h, 0d8h, 0dch, 076h, 000h, 000h, 000h, 000h
700 000150C1 D8DC7600000000      <1>
701 000150C8 000078CCCCCCD8CCC6- <1>     db  000h, 000h, 078h, 0cch, 0cch, 0cch, 0d8h, 0cch, 0c6h, 0c6h, 0c6h, 0cch, 000h, 000h, 000h, 000h
701 000150D1 C6C6CC00000000      <1>
702 000150D8 0000FEC6C6C0C0C0C0- <1>     db  000h, 000h, 0feh, 0c6h, 0c6h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h
702 000150E1 C0C0C000000000      <1>
703 000150E8 00000000FE6C6C6C6C- <1>     db  000h, 000h, 000h, 000h, 0feh, 06ch, 06ch, 06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h
703 000150F1 6C6C6C00000000      <1>
704 000150F8 000000FEC660301830- <1>     db  000h, 000h, 000h, 0feh, 0c6h, 060h, 030h, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h
704 00015101 60C6FE00000000      <1>
705 00015108 00000000007ED8D8D8- <1>     db  000h, 000h, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 0d8h, 0d8h, 070h, 000h, 000h, 000h, 000h
705 00015111 D8D87000000000      <1>
706 00015118 000000006666666666- <1>     db  000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0c0h, 000h, 000h, 000h
706 00015121 7C6060C0000000      <1>
707 00015128 0000000076DC181818- <1>     db  000h, 000h, 000h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
707 00015131 18181800000000      <1>
708 00015138 0000007E183C666666- <1>     db  000h, 000h, 000h, 07eh, 018h, 03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h
708 00015141 3C187E00000000      <1>
709 00015148 000000386CC6C6FEC6- <1>     db  000h, 000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h
709 00015151 C66C3800000000      <1>
710 00015158 0000386CC6C6C66C6C- <1>     db  000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 06ch, 06ch, 06ch, 06ch, 0eeh, 000h, 000h, 000h, 000h
710 00015161 6C6CEE00000000      <1>
711 00015168 00001E30180C3E6666- <1>     db  000h, 000h, 01eh, 030h, 018h, 00ch, 03eh, 066h, 066h, 066h, 066h, 03ch, 000h, 000h, 000h, 000h
711 00015171 66663C00000000      <1>
712 00015178 00000000007EDBDBDB- <1>     db  000h, 000h, 000h, 000h, 000h, 07eh, 0dbh, 0dbh, 0dbh, 07eh, 000h, 000h, 000h, 000h, 000h, 000h
712 00015181 7E000000000000      <1>
713 00015188 00000003067EDBDBF3- <1>     db  000h, 000h, 000h, 003h, 006h, 07eh, 0dbh, 0dbh, 0f3h, 07eh, 060h, 0c0h, 000h, 000h, 000h, 000h
713 00015191 7E60C000000000      <1>
714 00015198 00001C3060607C6060- <1>     db  000h, 000h, 01ch, 030h, 060h, 060h, 07ch, 060h, 060h, 060h, 030h, 01ch, 000h, 000h, 000h, 000h
714 000151A1 60301C00000000      <1>
715 000151A8 0000007CC6C6C6C6C6- <1>     db  000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
715 000151B1 C6C6C600000000      <1>
716 000151B8 00000000FE0000FE00- <1>     db  000h, 000h, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h
716 000151C1 00FE0000000000      <1>
717 000151C8 0000000018187E1818- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h
717 000151D1 0000FF00000000      <1>
718 000151D8 00000030180C060C18- <1>     db  000h, 000h, 000h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 000h, 07eh, 000h, 000h, 000h, 000h
718 000151E1 30007E00000000      <1>
719 000151E8 0000000C1830603018- <1>     db  000h, 000h, 000h, 00ch, 018h, 030h, 060h, 030h, 018h, 00ch, 000h, 07eh, 000h, 000h, 000h, 000h
719 000151F1 0C007E00000000      <1>
720 000151F8 00000E1B1B18181818- <1>     db  000h, 000h, 00eh, 01bh, 01bh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
720 00015201 18181818181818      <1>
721 00015208 1818181818181818D8- <1>     db  018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h, 0d8h, 070h, 000h, 000h, 000h, 000h
721 00015211 D8D87000000000      <1>
722 00015218 000000001818007E00- <1>     db  000h, 000h, 000h, 000h, 018h, 018h, 000h, 07eh, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h
722 00015221 18180000000000      <1>
723 00015228 000000000076DC0076- <1>     db  000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h
723 00015231 DC000000000000      <1>
724 00015238 00386C6C3800000000- <1>     db  000h, 038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
724 00015241 00000000000000      <1>
725 00015248 000000000000001818- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
725 00015251 00000000000000      <1>
726 00015258 000000000000000018- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
726 00015261 00000000000000      <1>
727 00015268 000F0C0C0C0C0CEC6C- <1>     db  000h, 00fh, 00ch, 00ch, 00ch, 00ch, 00ch, 0ech, 06ch, 06ch, 03ch, 01ch, 000h, 000h, 000h, 000h
727 00015271 6C3C1C00000000      <1>
728 00015278 00D86C6C6C6C6C0000- <1>     db  000h, 0d8h, 06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
728 00015281 00000000000000      <1>
729 00015288 0070D83060C8F80000- <1>     db  000h, 070h, 0d8h, 030h, 060h, 0c8h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
729 00015291 00000000000000      <1>
730 00015298 000000007C7C7C7C7C- <1>     db  000h, 000h, 000h, 000h, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 000h, 000h, 000h, 000h, 000h
730 000152A1 7C7C0000000000      <1>
731 000152A8 000000000000000000- <1>     db  000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
731 000152B1 00000000000000      <1>
732                              <1> vgafont14alt:
733 000152B8 1D000000002466FF66- <1>     db  01dh, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 022h
733 000152C1 24000000000022      <1>
734 000152C8 006363632200000000- <1>     db  000h, 063h, 063h, 063h, 022h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02bh, 000h
734 000152D1 00000000002B00      <1>
735 000152D8 0000181818FF181818- <1>     db  000h, 000h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 02dh, 000h, 000h
735 000152E1 000000002D0000      <1>
736 000152E8 00000000FF00000000- <1>     db  000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 04dh, 000h, 000h, 0c3h
736 000152F1 0000004D0000C3      <1>
737 000152F8 E7FFDBC3C3C3C3C300- <1>     db  0e7h, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h, 000h, 000h, 054h, 000h, 000h, 0ffh, 0dbh
737 00015301 0000540000FFDB      <1>
738 00015308 9918181818183C0000- <1>     db  099h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 056h, 000h, 000h, 0c3h, 0c3h, 0c3h
738 00015311 00560000C3C3C3      <1>
739 00015318 C3C3C3663C18000000- <1>     db  0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 057h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h
739 00015321 570000C3C3C3C3      <1>
740 00015328 DBDBFF666600000058- <1>     db  0dbh, 0dbh, 0ffh, 066h, 066h, 000h, 000h, 000h, 058h, 000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h
740 00015331 0000C3C3663C18      <1>
741 00015338 3C66C3C30000005900- <1>     db  03ch, 066h, 0c3h, 0c3h, 000h, 000h, 000h, 059h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h
741 00015341 00C3C3C3663C18      <1>
742 00015348 18183C0000005A0000- <1>     db  018h, 018h, 03ch, 000h, 000h, 000h, 05ah, 000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h, 030h, 061h
742 00015351 FFC3860C183061      <1>
743 00015358 C3FF0000006D000000- <1>     db  0c3h, 0ffh, 000h, 000h, 000h, 06dh, 000h, 000h, 000h, 000h, 000h, 0e6h, 0ffh, 0dbh, 0dbh, 0dbh
743 00015361 0000E6FFDBDBDB      <1>
744 00015368 DB0000007600000000- <1>     db  0dbh, 000h, 000h, 000h, 076h, 000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h
744 00015371 00C3C3C3663C18      <1>
745 00015378 000000770000000000- <1>     db  000h, 000h, 000h, 077h, 000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h
745 00015381 C3C3DBDBFF6600      <1>
746 00015388 000091000000006E3B- <1>     db  000h, 000h, 091h, 000h, 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h
746 00015391 1B7ED8DC770000      <1>
```

```
00015398 009B0018187EC3C0C0- <1>     db  000h, 09bh, 000h, 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h
000153A1 C37E1818000000      <1>
000153A8 9D0000C3663C18FF18- <1>     db  09dh, 000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 000h, 000h, 000h, 09eh
000153B1 FF18180000009E      <1>
000153B8 00FC66667C62666F66- <1>     db  000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 0f3h, 000h, 000h, 000h, 0f1h, 000h
000153C1 66F3000000F100      <1>
000153C8 00181818FF18181800- <1>     db  000h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 000h, 0ffh, 000h, 000h, 000h, 0f6h, 000h, 000h
000153D1 FF000000F60000      <1>
000153D8 18180000FF00001818- <1>     db  018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h
000153E1 00000000            <1>
                             <1> vgafont16alt:
000153E5 1D000000000002466FF- <1>    db  01dh, 000h, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h
000153EE 66240000000000      <1>
000153F5 003000003C66C3C3DB- <1>     db  000h, 030h, 000h, 000h, 03ch, 066h, 0c3h, 0c3h, 0dbh, 0dbh, 0c3h, 0c3h, 066h, 03ch, 000h, 000h
000153FE DBC3C3663C0000      <1>
00015405 00004D0000C3E7FFFF- <1>     db  000h, 000h, 04dh, 000h, 000h, 0c3h, 0e7h, 0ffh, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h
0001540E DBC3C3C3C3C300      <1>
00015415 000000540000FFDB99- <1>     db  000h, 000h, 000h, 054h, 000h, 000h, 0ffh, 0dbh, 099h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch
0001541E 1818181818183C      <1>
00015425 00000000560000C3C3- <1>     db  000h, 000h, 000h, 000h, 056h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch
0001542E C3C3C3C3C3663C      <1>
00015435 1800000000570000C3- <1>     db  018h, 000h, 000h, 000h, 000h, 057h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh
0001543E C3C3C3C3DBDBFF      <1>
00015445 666600000000580000- <1>     db  066h, 066h, 000h, 000h, 000h, 000h, 058h, 000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 03ch
0001544E C3C3663C18183C      <1>
00015455 66C3C3000000005900- <1>     db  066h, 0c3h, 0c3h, 000h, 000h, 000h, 000h, 059h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h
0001545E 00C3C3C3663C18      <1>
00015465 1818183C000000005A- <1>     db  018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 05ah, 000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h
0001546E 0000FFC3860C18      <1>
00015475 3060C1C3FF00000000- <1>     db  030h, 060h, 0c1h, 0c3h, 0ffh, 000h, 000h, 000h, 000h, 06dh, 000h, 000h, 000h, 000h, 000h, 0e6h
0001547E 6D000000000000E6    <1>
00015485 FFDBDBDBDBDB000000- <1>     db  0ffh, 0dbh, 0dbh, 0dbh, 0dbh, 0dbh, 000h, 000h, 000h, 000h, 076h, 000h, 000h, 000h, 000h, 000h
0001548E 00760000000000      <1>
00015495 C3C3C3C3663C180000- <1>     db  0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h, 077h, 000h, 000h, 000h, 000h
0001549E 00007700000000      <1>
000154A5 00C3C3C3DBDBFF6600- <1>     db  000h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h, 000h, 000h, 000h, 078h, 000h, 000h, 000h
000154AE 00000078000000      <1>
000154B5 0000C3663C183C66C3- <1>     db  000h, 000h, 0c3h, 066h, 03ch, 018h, 03ch, 066h, 0c3h, 000h, 000h, 000h, 000h, 091h, 000h, 000h
000154BE 00000000910000      <1>
000154C5 0000006E3B1B7ED8DC- <1>     db  000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h, 000h, 000h, 09bh, 000h
000154CE 77000000009B00      <1>
000154D5 18187EC3C0C0C0C37E- <1>     db  018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h, 09dh
000154DE 1818000000009D      <1>
000154E5 0000C3663C18FF18FF- <1>     db  000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h
000154EE 18181800000000      <1>
000154F5 9E00FC66667C62666F- <1>     db  09eh, 000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 066h, 0f3h, 000h, 000h, 000h
000154FE 666666F3000000      <1>
00015505 00AB00C0C0C2C6CC18- <1>     db  000h, 0abh, 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 060h, 0ceh, 09bh, 006h, 00ch, 01fh
0001550E 3060CE9B060C1F      <1>
00015515 0000AC00C0C0C2C6CC- <1>     db  000h, 000h, 0ach, 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 066h, 0ceh, 096h, 03eh, 006h
0001551E 183066CE963E06      <1>
00015525 06000000            <1>     db  006h, 000h, 000h, 000h

00015529 90                           align 2

                                      ; EPOCH Variables
                                      ; 13/04/2015 - Retro UNIX 386 v1 Beginning
                                      ; 09/04/2013 epoch variables
                                      ; Retro UNIX 8086 v1 Prototype: UNIXCOPY.ASM, 10/03/2013
                                      ;
0001552A B207                         year:        dw 1970
0001552C 0100                         month:       dw 1
0001552E 0100                         day:  dw 1
00015530 0000                         hour:        dw 0
00015532 0000                         minute: dw 0
00015534 0000                         second: dw 0

                                      DMonth:
00015536 0000                               dw 0
00015538 1F00                               dw 31
0001553A 3B00                               dw 59
0001553C 5A00                               dw 90
0001553E 7800                               dw 120
00015540 9700                               dw 151
00015542 B500                               dw 181
00015544 D400                               dw 212
00015546 F300                               dw 243
00015548 1101                               dw 273
0001554A 3001                               dw 304
0001554C 4E01                               dw 334

                                      ; 20/02/2017
                                      KERNELFSIZE  equ $  ; 04/07/2016

                                      bss_start:

                                      ABSOLUTE bss_start

0001554E <res 00000002>               alignb 8 ; 25/12/2016

                                            ; 15/04/2016
                                            ; TRDOS 386 (TRDOS v2.0)
                                            ;      80 interrupts
                                            ; 11/03/2015
                                            ; Interrupt Descriptor Table (20/08/2014)
                                      idt:
                                            ;resb  64*8 ; INT 0 to INT 3Fh
                                            ; 15/04/2016
00015550 <res 00000280>                     resb   80*8 ; INT 0 to INT 4Fh

                                      idt_end:

                                      ;alignb 4

                                      task_state_segment:
                                            ; 24/03/2015
```

```
000157D0 <res 00000002>         tss.link:   resw 1
000157D2 <res 00000002>                   resw 1
                                ; tss offset 4
000157D4 <res 00000004>         tss.esp0:   resd 1
000157D8 <res 00000002>         tss.ss0:    resw 1
000157DA <res 00000002>                   resw 1
000157DC <res 00000004>         tss.esp1:   resd 1
000157E0 <res 00000002>         tss.ss1:    resw 1
000157E2 <res 00000002>                   resw 1
000157E4 <res 00000004>         tss.esp2:   resd 1
000157E8 <res 00000002>         tss.ss2:    resw 1
000157EA <res 00000002>                   resw 1
                                ; tss offset 28
000157EC <res 00000004>         tss.CR3:    resd 1
000157F0 <res 00000004>         tss.eip:    resd 1
000157F4 <res 00000004>         tss.eflags: resd 1
                                ; tss offset 40
000157F8 <res 00000004>         tss.eax:    resd 1
000157FC <res 00000004>         tss.ecx:    resd 1
00015800 <res 00000004>         tss.edx:    resd 1
00015804 <res 00000004>         tss.ebx:    resd 1
00015808 <res 00000004>         tss.esp:    resd 1
0001580C <res 00000004>         tss.ebp:    resd 1
00015810 <res 00000004>         tss.esi:    resd 1
00015814 <res 00000004>         tss.edi:    resd 1
                                ; tss offset 72
00015818 <res 00000002>         tss.ES:     resw 1
0001581A <res 00000002>                   resw 1
0001581C <res 00000002>         tss.CS:         resw 1
0001581E <res 00000002>                   resw 1
00015820 <res 00000002>         tss.SS:         resw 1
00015822 <res 00000002>                   resw 1
00015824 <res 00000002>         tss.DS:         resw 1
00015826 <res 00000002>                   resw 1
00015828 <res 00000002>         tss.FS:         resw 1
0001582A <res 00000002>                   resw 1
0001582C <res 00000002>         tss.GS:         resw 1
0001582E <res 00000002>                   resw 1
00015830 <res 00000002>         tss.LDTR:   resw 1
00015832 <res 00000002>                   resw 1
                                ; tss offset 100
00015834 <res 00000002>                   resw 1
00015836 <res 00000002>         tss.IOPB:   resw 1
                                ; tss offset 104
                                tss_end:

00015838 <res 00000004>         k_page_dir:  resd 1 ; Kernel's (System) Page Directory address
                                                 ; (Physical address = Virtual address)
0001583C <res 00000004>         memory_size: resd 1 ; memory size in pages
00015840 <res 00000004>         free_pages:  resd 1 ; number of free pages
00015844 <res 00000004>         next_page:   resd 1 ; offset value in M.A.T. for
                                                 ; first free page search
00015848 <res 00000004>         last_page:   resd 1 ; offset value in M.A.T. which
                                                 ; next free page search will be
                                                 ; stopped after it. (end of M.A.T.)
0001584C <res 00000004>         first_page:  resd 1 ; offset value in M.A.T. which
                                                 ; first free page search
                                                 ; will be started on it. (for user)
00015850 <res 00000004>         mat_size:    resd 1 ; Memory Allocation Table size in pages

                                ; 02/09/2014 (Retro UNIX 386 v1)
                                ; 04/12/2013 (Retro UNIX 8086 v1)
00015854 <res 00000002>         CRT_START:   resw 1         ; starting address in regen buffer
                                                      ; NOTE: active page only
00015856 <res 00000010>         CURSOR_POSN: resw 8 ; cursor positions for video pages
                                ACTIVE_PAGE:
00015866 <res 00000001>         ptty:            resb 1 ; current tty
                                ; 01/07/2015 - 29/01/2016
00015867 <res 00000001>         ccolor:          resb 1 ; current color attribute
                                ; 26/10/2015
                                ; 07/09/2014
00015868 <res 00000014>         ttychr:      resw ntty+2 ; Character buffer (multiscreen)

                                ; 18/05/2015 (03/06/2013 - Retro UNIX 8086 v1 feature only!)
0001587C <res 00000004>         p_time:      resd 1     ; present time (for systime & sysmdate)

                                ; 18/05/2015 (16/08/2013 - Retro UNIX 8086 v1 feature only !)
                                ; (open mode locks for pseudo TTYs)
                                ; [ major tty locks (return error in any conflicts) ]
00015880 <res 00000014>         ttyl:        resw ntty+2 ; opening locks for TTYs.

                                ; 15/04/2015 (Retro UNIX 386 v1)
                                ; 22/09/2013 (Retro UNIX 8086 v1)
00015894 <res 0000000A>         wlist:       resb ntty+2 ; wait channel list (0 to 9 for TTYs)
                                ; 15/04/2015 (Retro UNIX 386 v1)
                                ;; 12/07/2014 -> sp_init set comm. parameters as 0E3h
                                ;; 0 means serial port is not available
                                ;;comprm: ; 25/06/2014
0001589E <res 00000001>         com1p:       resb 1  ;;0E3h
0001589F <res 00000001>         com2p:       resb 1  ;;0E3h

                                ; 17/11/2015
                                ; request for response (from the terminal)
000158A0 <res 00000002>         req_resp:    resw 1
                                ; 07/11/2015
000158A2 <res 00000001>         ccomport:    resb 1 ; current COM (serial) port
                                                 ; (0= COM1, 1= COM2)
                                ; 09/11/2015
000158A3 <res 00000001>         comqr:           resb 1 ; 'query or response' sign (u9.s, 'sndc')
                                ; 07/11/2015
000158A4 <res 00000002>         rchar:           resw 1 ; last received char for COM 1 and COM 2
000158A6 <res 00000002>         schar:           resw 1 ; last sent char for COM 1 and COM 2
```

```
                                 ; 22/08/2014 (RTC)
                                 ; (Packed BCD)
000158A8 <res 00000001>          time_seconds: resb 1
000158A9 <res 00000001>          time_minutes: resb 1
000158AA <res 00000001>          time_hours:   resb 1
000158AB <res 00000001>          date_wday:    resb 1
000158AC <res 00000001>          date_day:     resb 1
000158AD <res 00000001>          date_month:   resb 1
000158AE <res 00000001>          date_year:    resb 1
000158AF <res 00000001>          date_century: resb 1

                                 ; 24/01/2016
000158B0 <res 00000004>          RTC_LH:            resd 1
000158B4 <res 00000001>          RTC_WAIT_FLAG: resb 1
000158B5 <res 00000001>          USER_FLAG:     resb 1
                                 ; 19/05/2016
                                 ;RTC_second:
000158B6 <res 00000001>          RTC_2Hz:       resb 1 ;  from 2Hz interrupt to 1Hz timer event function

                                 %include 'diskbss.s'      ; UNINITIALIZED DISK (BIOS) DATA
                             <1> ; ****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - diskbss.s
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 24/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 24/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Turkish Rational DOS
                             <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
                             <1> ;
                             <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                             <1> ; diskbss.inc (10/07/2015)
                             <1> ;
                             <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)
                             <1> ; ****************************************************************************
                             <1>
                             <1> ; Retro UNIX 386 v1 Kernel - DISKBSS.INC
                             <1> ; Last Modification: 10/07/2015
                             <1> ;     (Unnitialized Disk Parameters Data section for 'DISKIO.INC')
                             <1>
000158B7 <res 00000001>      <1> alignb 2
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     TIMER DATA AREA           :
                             <1> ;----------------------------------------
                             <1>
                             <1> TIMER_LH:    ; 16/02/205
000158B8 <res 00000002>      <1> TIMER_LOW:      resw      1               ; LOW WORD OF TIMER COUNT
000158BA <res 00000002>      <1> TIMER_HIGH:     resw      1               ; HIGH WORD OF TIMER COUNT
000158BC <res 00000001>      <1> TIMER_OFL:      resb      1               ; TIMER HAS ROLLED OVER SINCE LAST READ
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     DISKETTE DATA AREAS       :
                             <1> ;----------------------------------------
                             <1>
000158BD <res 00000001>      <1> SEEK_STATUS: resb   1
000158BE <res 00000001>      <1> MOTOR_STATUS:      resb   1
000158BF <res 00000001>      <1> MOTOR_COUNT: resb   1
000158C0 <res 00000001>      <1> DSKETTE_STATUS:    resb   1
000158C1 <res 00000007>      <1> NEC_STATUS:  resb   7
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     ADDITIONAL MEDIA DATA            :
                             <1> ;----------------------------------------
                             <1>
000158C8 <res 00000001>      <1> LASTRATE:    resb   1
000158C9 <res 00000001>      <1> HF_STATUS:   resb   1
000158CA <res 00000001>      <1> HF_ERROR:    resb   1
000158CB <res 00000001>      <1> HF_INT_FLAG: resb   1
000158CC <res 00000001>      <1> HF_CNTRL:    resb   1
000158CD <res 00000004>      <1> DSK_STATE:   resb   4
000158D1 <res 00000002>      <1> DSK_TRK:     resb   2
                             <1>
                             <1> ;----------------------------------------
                             <1> ;     FIXED DISK DATA AREAS            :
                             <1> ;----------------------------------------
                             <1>
000158D3 <res 00000001>      <1> DISK_STATUS1:      resb   1            ; FIXED DISK STATUS
000158D4 <res 00000001>      <1> HF_NUM:            resb   1            ; COUNT OF FIXED DISK DRIVES
000158D5 <res 00000001>      <1> CONTROL_BYTE:      resb   1            ; HEAD CONTROL BYTE
                             <1> ;@PORT_OFF   resb   1            ; RESERVED (PORT OFFSET)
                             <1> ;port1_off   resb   1            ; Hard disk controller 1 - port offset
                             <1> ;port2_off   resb   1            ; Hard idsk controller 2 - port offset
                             <1>
000158D6 <res 00000002>      <1> alignb 4
                             <1>
                             <1> ;HF_TBL_VEC: resd   1            ; Primary master disk param. tbl. pointer
                             <1> ;HF1_TBL_VEC:      resd   1            ; Primary slave disk param. tbl. pointer
                             <1> HF_TBL_VEC: ; 22/12/2014
000158D8 <res 00000004>      <1> HDPM_TBL_VEC:      resd   1            ; Primary master disk param. tbl. pointer
000158DC <res 00000004>      <1> HDPS_TBL_VEC:      resd   1            ; Primary slave disk param. tbl. pointer
000158E0 <res 00000004>      <1> HDSM_TBL_VEC:      resd   1            ; Secondary master disk param. tbl. pointer
000158E4 <res 00000004>      <1> HDSS_TBL_VEC:      resd   1            ; Secondary slave disk param. tbl. pointer
                             <1>
                             <1> ; 03/01/2015
000158E8 <res 00000001>      <1> LBAMode:           resb   1
                             <1>
                             <1> ; *****************************************************************************

                                 ;;; Real Mode Data (10/07/2015 - BSS)

                                 ;alignb 2
                                 
                                 ; 10/01/2016
                                 %include 'trdoskx.s'      ; UNINITIALIZED KERNEL (Logical Drive & FS) DATA
                             <1> ; ****************************************************************************
                             <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - UNINITIALIZED DATA : trdoskx.s
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Last Update: 28/08/2017
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Beginning: 04/01/2016
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Assembler: NASM version 2.11 (trdos386.s)
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
                             <1> ; TRDOS2.ASM (09/11/2011)
                             <1> ; ****************************************************************************
                             <1> ; DRV_INIT.ASM [26/09/2009] Last Update: 07/08/2011
                             <1> ; MAINPROG.ASM [17/01/2004] Last Update: 09/11/2011
                             <1> ; DIR.ASM      [17/01/2004] Last Update: 09/10/2011
                             <1> ; CMD_INTR.ASM [29/01/2005] Last update: 09/11/2011
                             <1> ; DRV_FAT.ASM  [07/07/2009] Last update: 21/08/2011
                             <1> 
000158E9 <res 00000003>      <1> alignb 4
                             <1> 
                             <1> ; MAINPROG.ASM
000158EC <res 00000004>      <1> MainProgCfg_FileSize:   resd 1 ; 14/04/2016
000158F0 <res 00000004>      <1> MainProgCfg_LineOffset: resd 1 ; 14/04/2016
                             <1> 
000158F4 <res 00000004>      <1> Current_VolSerial: resd 1
                             <1> 
000158F8 <res 00000004>      <1> Current_Dir_FCluster: resd 1
                             <1> 
000158FC <res 00000001>      <1> Current_Dir_Level: resb 1
000158FD <res 00000001>      <1> Current_FATType: resb 1
000158FE <res 00000001>      <1> Current_Drv: resb 1
000158FF <res 00000001>      <1> Current_Dir_Drv:   resb 1 ; '?'
00015900 <res 00000001>      <1>                    resb 1 ; ':'
00015901 <res 00000001>      <1> Current_Dir_Root:  resb 1 ; '/'
00015902 <res 0000005A>      <1> Current_Directory: resb 90
0001595C <res 00000001>      <1> End_Of_Current_Dir_Str: resb 1
0001595D <res 00000001>      <1> Current_Dir_StrLen: resb 1
                             <1> 
0001595E <res 00000001>      <1> CursorColumn:      resb 1
0001595F <res 00000001>      <1> CmdArgStart:    resb 1
                             <1> 
                             <1> ; 03/02/2016
00015960 <res 0000004E>      <1> Remark:            resb 78
                             <1> 
000159AE <res 00000050>      <1> CommandBuffer:     resb 80
                             <1> 
000159FE <res 00000100>      <1> TextBuffer:  resb 256
                             <1> 
                             <1> MasterBootBuff:
00015AFE <res 000001BE>      <1> MasterBootCode: resb 1BEh
00015CBC <res 00000040>      <1> PartitionTable: resb 64
00015CFC <res 00000002>      <1> MBIDCode: resw 1
                             <1> 
                             <1> PTable_Buffer:
00015CFE <res 00000040>      <1> PTable_hd0: resb 64
00015D3E <res 00000040>      <1> PTable_hd1: resb 64
00015D7E <res 00000040>      <1> PTable_hd2: resb 64
00015DBE <res 00000040>      <1> PTable_hd3: resb 64
00015DFE <res 00000040>      <1> PTable_ep0: resb 64
00015E3E <res 00000040>      <1> PTable_ep1: resb 64
00015E7E <res 00000040>      <1> PTable_ep2: resb 64
00015EBE <res 00000040>      <1> PTable_ep3: resb 64
                             <1> 
00015EFE <res 00000001>      <1> scount:      resb 1 ; 16/05/2016 (diskio.s, 'int33h:')
00015EFF <res 00000001>      <1> HD_LBA_yes: resb 1
00015F00 <res 00000001>      <1> PP_Counter: resb 1
00015F01 <res 00000001>      <1> EP_Counter: resb 1
                             <1> 
00015F02 <res 00000004>      <1> EP_StartSector: resd 1
00015F06 <res 00000004>      <1>                 resd 1
00015F0A <res 00000004>      <1>                 resd 1
00015F0E <res 00000004>      <1>                 resd 1
                             <1> 
00015F12 <res 00000200>      <1> DOSBootSectorBuff: resb 512
                             <1> 
                             <1> FAT_BuffDescriptor:
00016112 <res 00000004>      <1> FAT_CurrentCluster: resd 1
00016116 <res 00000001>      <1> FAT_BuffValidData: resb 1
00016117 <res 00000001>      <1> FAT_BuffDrvName: resb 1
00016118 <res 00000002>      <1> FAT_BuffOffset: resw 1
0001611A <res 00000004>      <1> FAT_BuffSector: resd 1
                             <1> 
0001611E <res 00000004>      <1> FAT_ClusterCounter: resd 1
00016122 <res 00000004>      <1> LastCluster: resd 1
                             <1> 
                             <1> ; 16/05/2016
                             <1> ;; 18/03/2016 (TRDOS v2.0)
                             <1> ;ClusterBuffer_Valid: resb 1
                             <1> 
                             <1> Dir_BuffDescriptor:
00016126 <res 00000001>      <1> DirBuff_DRV: resb 1
00016127 <res 00000001>      <1> DirBuff_FATType: resb 1
00016128 <res 00000001>      <1> DirBuff_ValidData: resb 1
00016129 <res 00000002>      <1> DirBuff_CurrentEntry: resw 1
0001612B <res 00000002>      <1> DirBuff_LastEntry: resw 1
0001612D <res 00000004>      <1> DirBuff_Cluster: resd 1
00016131 <res 00000002>      <1> DirBuffer_Size: resw 1
                             <1> ;DirBuff_EntryCounter: resw 1
                             <1> 
```

```
                              <1> ; 01/02/2016
                              <1> ; these are on (real mode) segment 8000h and later
                              <1> ; FAT_Buffer:      resb 1536 ; 3 sectors
                              <1> ; Dir_Buffer:      resb 512*32
                              <1> ; Logical_DOSDisks:  resb 6656 ; 26 * 256 bytes
                              <1>
                              <1> ; 18/01/2016
                              <1>
00016133 <res 00000004>       <1> FreeClusterCount: resd 1
                              <1>
00016137 <res 00000004>       <1> VolSize_Unit1:   resd 1
0001613B <res 00000004>       <1> VolSize_Unit2:   resd 1
                              <1>
0001613F <res 00000004>       <1> Vol_Tot_Sec_Str_Start:       resd 1
00016143 <res 0000000A>       <1> Vol_Tot_Sec_Str:       resb 10
0001614D <res 00000001>       <1> Vol_Tot_Sec_Str_End:         resb 1
0001614E <res 00000001>       <1> resb 1
0001614F <res 00000004>       <1> Vol_Free_Sectors_Str_Start: resd 1
00016153 <res 0000000A>       <1> Vol_Free_Sectors_Str:        resb 10
0001615D <res 00000001>       <1> Vol_Free_Sectors_Str_End:   resb 1
                              <1>
                              <1> ; 10/02/2016
0001615E <res 00000001>       <1> RUN_CDRV: resb 1 ; CMD_INTR.ASM  ; 09/11/2011
                              <1>
                              <1> ; 24/01/2016
0001615F <res 00000080>       <1> PATH_Array:     resb 128 ; DIR.ASM ; 09/10/2011
                              <1> ; 06/02/2016
000161DF <res 00000004>       <1> CCD_DriveDT: resd 1 ; DIR.ASM ; (word)
000161E3 <res 00000001>       <1> CCD_Level:   resb 1 ; DIR.ASM
000161E4 <res 00000001>       <1> Last_Dir_Level:    resb 1 ; DIR.ASM
                              <1> ;
000161E5 <res 00000002>       <1> CDLF_AttributesMask: resw 1 ; DIR.ASM
000161E7 <res 00000004>       <1> CDLF_FNAddress:    resd 1 ; DIR.ASM (word)
000161EB <res 00000002>       <1> CDLF_DEType: resw 1 ; DIR.ASM
                              <1> ;
000161ED <res 00000001>       <1> CD_COMMAND:  resb 1 ; DIR.ASM
                              <1>
000161EE <res 00000002>       <1> alignb 4
                              <1>
                              <1> ; 29/01/2016
000161F0 <res 00000001>       <1> Program_Exit:      resb 1 ; CMD_INTR.ASM  ; 09/11/2011
                              <1>
                              <1> ;alignb 4
                              <1> ; 23/02/2016
000161F1 <res 00000001>       <1> disk_rw_op:  resb 1 ;  0 = disk read, 1 = disk write
                              <1> ;disk_rw_spt:      resb 1 ; sectors per track (<= 63) /// (<256)
                              <1> ; 31/01/2016
000161F2 <res 00000001>       <1> retry_count:       resb 1 ; DISK_IO.ASM ; 20/07/2011 (CHS_RetryCount)
000161F3 <res 00000001>       <1> disk_rw_err:       resb 1 ; DISK_IO.ASM ; (Disk_IO_err_code)
000161F4 <res 00000004>       <1> sector_count:      resd 1 ; DISK_IO.ASM ; (Disk_RW_SectorCount)
                              <1>
                              <1> ; 06/02/2016 (long name)
000161F8 <res 00000002>       <1> FDE_AttrMask:         resw 1 ; DIR.ASM
000161FA <res 00000002>       <1> AmbiguousFileName: resw 1 ; DIR.ASM
000161FC <res 00000001>       <1> PreviousAttr:         resb 1 ; DIR.ASM
                              <1> ;
000161FD <res 00000001>       <1> LongNameFound:   resb 1     ; DIR.ASM
000161FE <res 00000001>       <1> LFN_EntryLength: resb 1   ; DIR.ASM
000161FF <res 00000001>       <1> LFN_CheckSum:    resb 1   ; DIR.ASM
00016200 <res 00000084>       <1> LongFileName:    resb 132 ; DIR.ASM
                              <1>
                              <1> ;PATH_Array_Ptr: resw 1 ; DIR.ASM
00016284 <res 00000001>       <1> PATH_CDLevel:       resb 1 ; DIR.ASM
00016285 <res 00000001>       <1> PATH_Level:   resb 1 ; DIR.ASM
                              <1>
                              <1> ; 07/02/2016
00016286 <res 0000000D>       <1> Dir_File_Name:     resb 13 ; DIR.ASM ; 09/10/2011
                              <1>
                              <1> ; 10/02/2016
00016293 <res 0000000D>       <1> Dir_Entry_Name:    resb 13 ; DIR.ASM
                              <1>
                              <1> alignb 2
                              <1>
000162A0 <res 00000002>       <1> AttributesMask: resw 1 ; CMD_INTR.ASM ; 09/11/2011
                              <1>
                              <1> ; 10/02/2016 (128 bytes -> 126 bytes)
                              <1> ; 08/02/2016
                              <1> ;FFF Structure (128 bytes) ; DIR.ASM ; 09/10/2011
000162A2 <res 00000001>       <1> FindFile_Drv:               resb 1
000162A3 <res 00000041>       <1> FindFile_Directory:         resb 65
000162E4 <res 0000000D>       <1> FindFile_Name:              resb 13
                              <1> FindFile_LongNameEntryLength:
000162F1 <res 00000001>       <1> FindFile_LongNameYes:       resb 1 ; Sign for longname procedures
                              <1> ;Above 80 bytes form
                              <1> ;TR-DOS Source/Destination File FullName Format/Structure
000162F2 <res 00000002>       <1> FindFile_AttributesMask:  resw 1
000162F4 <res 00000020>       <1> FindFile_DirEntry:   resb 32
00016314 <res 00000004>       <1> FindFile_DirFirstCluster: resd 1
00016318 <res 00000004>       <1> FindFile_DirCluster:        resd 1
0001631C <res 00000002>       <1> FindFile_DirEntryNumber:  resw 1
0001631E <res 00000002>       <1> FindFile_MatchCounter:      resw 1
00016320 <res 00000002>       <1> FindFile_Reserved:   resw 1 ; 06/03/2016
                              <1>
00016322 <res 00000004>       <1> First_Path_Pos: resd 1    ; DIR.ASM ; 09/10/2011
00016326 <res 00000004>       <1> Last_Slash_Pos: resd 1    ; DIR.ASM
                              <1>
                              <1> ; 10/02/2016
0001632A <res 00000002>       <1> File_Count:     resw 1    ; DIR.ASM ; 09/10/2011
0001632C <res 00000002>       <1> Dir_Count:      resw 1
0001632E <res 00000004>       <1> Total_FSize:    resd 1
00016332 <res 00000004>       <1> TFS_Dec_Begin:  resd 1
00016336 <res 0000000A>       <1>                 resb 10
00016340 <res 00000001>       <1> TFS_Dec_End:    resb 1
```

```
                             <1>
00016341 <res 00000001>      <1> PrintDir_RowCounter: resb 1
                             <1>
00016342 <res 00000002>      <1> alignb 4
                             <1> ; 15/02/2015 ('show' command variables)
00016344 <res 00000004>      <1> Show_FDT:    resd 1
00016348 <res 00000004>      <1> Show_LDDDT:  resd 1
0001634C <res 00000004>      <1> Show_Cluster:      resd 1
00016350 <res 00000004>      <1> Show_FileSize:     resd 1
00016354 <res 00000004>      <1> Show_FilePointer: resd 1
00016358 <res 00000002>      <1> Show_ClusterPointer: resw 1
0001635A <res 00000002>      <1> Show_ClusterSize: resw 1
0001635C <res 00000001>      <1> Show_RowCount:     resb 1
                             <1>
0001635D <res 00000003>      <1> alignb 4
                             <1> ; 21/02/2016
00016360 <res 00000004>      <1> DelFile_FNPointer: resd 1 ; ; CMD_INTR.ASM (word) ; 09/11/2011
                             <1> ; 27/02/2016
                             <1> ; DIR.ASM (09/10/2011)
00016364 <res 00000004>      <1> DelFile_FCluster:  resd 1
00016368 <res 00000002>      <1> DelFile_EntryCounter:     resw 1
0001636A <res 00000001>      <1> DelFile_LNEL:             resb 1
0001636B <res 00000001>      <1> resb 1
                             <1>
                             <1> ; DIR.ASM
0001636C <res 00000004>      <1> mkdir_DirName_Offset:     resd 1
00016370 <res 00000004>      <1> mkdir_FFCluster:   resd 1
00016374 <res 00000004>      <1> mkdir_LastDirCluster:     resd 1
00016378 <res 00000004>      <1> mkdir_FreeSectors: resd 1
0001637C <res 00000002>      <1> mkdir_attrib:             resw 1
0001637E <res 00000001>      <1> mkdir_SecPerClust: resb 1
0001637F <res 00000001>      <1> mkdir_add_new_cluster:    resb 1
00016380 <res 0000000D>      <1> mkdir_Name:        resb 13
0001638D <res 00000002>      <1> resw 1 ; 01/03/2016
                             <1> ; 27/02/2016
0001638F <res 00000001>      <1> RmDir_MultiClusters:      resb 1
00016390 <res 00000004>      <1> RmDir_DirEntryOffset:     resd 1 ; 01/03/2016 (word -> dword)
00016394 <res 00000004>      <1> RmDir_ParentDirCluster: resd 1
00016398 <res 00000004>      <1> RmDir_DirLastCluster:   resd 1
0001639C <res 00000004>      <1> RmDir_PreviousCluster:  resd 1
                             <1> ; 22/02/2016
000163A0 <res 00000001>      <1> UPDLMDT_CDirLevel: resb 1
000163A1 <res 00000004>      <1> UPDLMDT_CDirFCluster:     resd 1
                             <1>
000163A5 <res 00000003>      <1> alignb 4
                             <1> ; DRV_FAT.ASM ; 21/08/2011
000163A8 <res 00000004>      <1> gffc_next_free_cluster:  resd 1
000163AC <res 00000004>      <1> gffc_first_free_cluster: resd 1
000163B0 <res 00000004>      <1> gffc_last_free_cluster:  resd 1
                             <1>
                             <1> ;29/04/2016
                             <1> Cluster_Index: ; resd 1
                             <1> ; 22/02/2016
000163B4 <res 00000004>      <1> ClusterValue:      resd 1
                             <1> ; 04/03/2016
000163B8 <res 00000001>      <1> Attributes:  resb 1
                             <1> ;;CFS_error:  resb 1 ;; 01/03/2016
000163B9 <res 00000001>      <1> resb 1
000163BA <res 00000001>      <1> CFS_OPType: resb 1
000163BB <res 00000001>      <1> CFS_Drv:    resb 1
000163BC <res 00000004>      <1> CFS_CC:          resd 1
000163C0 <res 00000004>      <1> CFS_FAT32FSINFOSEC: resd 1
000163C4 <res 00000004>      <1> CFS_FAT32FC: resd 1
                             <1>
                             <1> ; 27/02/2016
                             <1> ;alignb 4
000163C8 <res 00000004>      <1> glc_prevcluster: resd 1 ; DRV_FAT.ASM (21/08/2011)
                             <1> ; 22/10/2016
000163CC <res 00000004>      <1> glc_index:   resd 1 ;  Last Cluster Index (22/10/2016)
                             <1>
                             <1> ; DIR.ASM
000163D0 <res 00000002>      <1> DLN_EntryNumber: resw 1
000163D2 <res 00000001>      <1> DLN_40h:      resb 1
                             <1> ; 28/02/2016
000163D3 <res 00000001>      <1> TCC_FATErr:   resb 1 ; DRV_FAT.ASM
                             <1>
                             <1> alignb 4
                             <1> ; DIR.ASM (09/10/2011)
000163D4 <res 00000002>      <1> LCDE_EntryIndex: resw 1 ; LCDE_EntryOffset
000163D6 <res 00000002>      <1> LCDE_ClusterSN:  resw 1
000163D8 <res 00000004>      <1> LCDE_Cluster:       resd 1
000163DC <res 00000004>      <1> LCDE_ByteOffset: resd 1
                             <1>
                             <1> ;alignb4
                             <1> ; 06/03/2016 (word -> dword)
                             <1> ; CMD_INTR.ASM (01/08/2010)
000163E0 <res 00000004>      <1> SourceFilePath:         resd 1
000163E4 <res 00000004>      <1> DestinationFilePath: resd 1
                             <1>
                             <1> ;alignb 4
                             <1> ; 06/03/2016
                             <1> ; FILE.ASM (09/10/2011)
                             <1> ;Source File Structure (same with 'Find File' Structure)
000163E8 <res 00000001>      <1> SourceFile_Drv:                 resb 1
000163E9 <res 00000041>      <1> SourceFile_Directory:           resb 65
0001642A <res 0000000D>      <1> SourceFile_Name:          resb 13
                             <1> SourceFile_LongNameEntryLength:
00016437 <res 00000001>      <1> SourceFile_LongNameYes:         resb 1 ; Sign for longname procedures
                             <1> ;Above 80 bytes
                             <1> ;is TR-DOS Source File FullName Format/Structure
00016438 <res 00000002>      <1> SourceFile_AttributesMask:      resw 1
0001643A <res 00000020>      <1> SourceFile_DirEntry:            resb 32
0001645A <res 00000004>      <1> SourceFile_DirFirstCluster:     resd 1
```

```
0001645E <res 00000004>      <1> SourceFile_DirCluster:          resd 1
00016462 <res 00000002>      <1> SourceFile_DirEntryNumber:      resw 1
00016464 <res 00000002>      <1> SourceFile_MatchCounter:  resw 1
                             <1> ; 16/03/2016
00016466 <res 00000001>      <1> SourceFile_SecPerClust:         resb 1
00016467 <res 00000001>      <1> SourceFile_Reserved:            resb 1
                             <1> ; Above is 128 bytes
                             <1>
                             <1> ;Destination File Structure (same with 'Find File' Structure)
00016468 <res 00000001>      <1> DestinationFile_Drv:            resb 1
00016469 <res 00000041>      <1> DestinationFile_Directory:      resb 65
000164AA <res 0000000D>      <1> DestinationFile_Name:           resb 13
                             <1> DestinationFile_LongNameEntryLength:
000164B7 <res 00000001>      <1> DestinationFile_LongNameYes:    resb 1 ; Sign for longname procedures
                             <1> ;Above 80 bytes
                             <1> ;is TR-DOS Destination File FullName Format/Structure
000164B8 <res 00000002>      <1> DestinationFile_AttributesMask: resw 1
000164BA <res 00000020>      <1> DestinationFile_DirEntry: resb 32
000164DA <res 00000004>      <1> DestinationFile_DirFirstCluster: resd 1
000164DE <res 00000004>      <1> DestinationFile_DirCluster:     resd 1
000164E2 <res 00000002>      <1> DestinationFile_DirEntryNumber: resw 1
000164E4 <res 00000002>      <1> DestinationFile_MatchCounter:   resw 1
                             <1> ; 16/03/2016
000164E6 <res 00000001>      <1> DestinationFile_SecPerClust:    resb 1
000164E7 <res 00000001>      <1> DestinationFile_Reserved: resb 1
                             <1> ; Above is 128 bytes
                             <1>
                             <1> ; 24/04/2016
000164E8 <res 00000002>      <1> resw 1
                             <1>
                             <1> ; 10/03/2016
                             <1> ; FILE.ASM
000164EA <res 00000001>      <1> move_cmd_phase:       resb 1
000164EB <res 00000001>      <1> msftdf_sf_df_drv:  resb 1
000164EC <res 00000004>      <1> msftdf_drv_offset: resd 1
                             <1>
                             <1> ; 11/03/2016
                             <1> ; DRV_FAT.ASM (21/08/2011)
000164F0 <res 00000004>      <1> FAT_anc_LCluster:  resd 1
000164F4 <res 00000004>      <1> FAT_anc_FFCluster: resd 1
                             <1>
                             <1> ;alignb 4
                             <1>
                             <1> ; 14/03/2016
                             <1> ; TRDOS 386 = TRDOS v2.0 feature only !
                             <1> ; 'allocate_memory_block' in 'memory.s'
000164F8 <res 00000004>      <1> mem_ipg_count:     resd 1 ; page count (for contiguous allocation)
000164FC <res 00000004>      <1> mem_pg_count:      resd 1 ; page count (for count down)
00016500 <res 00000004>      <1> mem_aperture:      resd 1 ; contiguous free pages (current)
00016504 <res 00000004>      <1> mem_max_aperture: resd 1 ; maximum value of contiguous free pages
00016508 <res 00000004>      <1> mem_pg_pos:  resd 1 ; mem. position (page #) of current aperture
0001650C <res 00000004>      <1> mem_max_pg_pos: resd 1 ; mem. position (page #) of max. aperture
                             <1>
                             <1> ; 15/03/2016
                             <1> ; FILE.ASM ('copy_source_file_to_destination_file')
00016510 <res 00000001>      <1> copy_cmd_phase:       resb 1
00016511 <res 00000001>      <1> csftdf_rw_err:           resb 1
00016512 <res 00000001>      <1> DestinationFileFound: resb 1
00016513 <res 00000001>      <1> csftdf_cdrv:             resb 1
00016514 <res 00000004>      <1> csftdf_filesize:      resd 1
                             <1> ; TRDOS386 (TRDOS v2.0)
00016518 <res 00000004>      <1> csftdf_sf_mem_addr:   resd 1
0001651C <res 00000004>      <1> csftdf_sf_mem_bsize:  resd 1
                             <1> ;
                             <1>
00016520 <res 00000004>      <1> csftdf_sf_cluster:    resd 1 ; 16/03/2016
00016524 <res 00000004>      <1> csftdf_df_cluster:    resd 1
                             <1> ; 16/03/2016
00016528 <res 00000004>      <1> csftdf_r_size:        resd 1
0001652C <res 00000004>      <1> csftdf_w_size:        resd 1
00016530 <res 00000004>      <1> csftdf_sf_rbytes:     resd 1
00016534 <res 00000004>      <1> csftdf_df_wbytes:     resd 1
00016538 <res 00000001>      <1> csftdf_percentage:    resb 1
                             <1> ; 17/03/2016
00016539 <res 00000001>      <1> csftdf_videopage:     resb 1
0001653A <res 00000002>      <1> csftdf_cursorpos:     resw 1
0001653C <res 00000004>      <1> csftdf_sf_drv_dt:     resd 1
00016540 <res 00000004>      <1> csftdf_df_drv_dt:     resd 1
                             <1>
                             <1> ; 21/03/2016
                             <1> ; 20/03/2016
                             <1> ; FILE.ASM
00016544 <res 00000004>      <1> createfile_Name_Offset:  resd 1
00016548 <res 00000004>      <1> createfile_FreeSectors:  resd 1
0001654C <res 00000004>      <1> createfile_size:         resd 1
00016550 <res 00000004>      <1> createfile_FFCluster:    resd 1 ; 11/03/2016
00016554 <res 00000004>      <1> createfile_LastDirCluster: resd 1
00016558 <res 00000004>      <1> createfile_Cluster:      resd 1
0001655C <res 00000004>      <1> createfile_PCluster:     resd 1
00016560 <res 00000001>      <1> createfile_attrib:  resb 1
00016561 <res 00000001>      <1> createfile_SecPerClust:  resb 1
00016562 <res 00000002>      <1> createfile_DirIndex:     resw 1
00016564 <res 00000004>      <1> createfile_CCount:  resd 1
00016568 <res 00000002>      <1> createfile_BytesPerSec:    resw 1 ; 23/03/2016
0001656A <res 00000001>      <1> createfile_wfc:              resb 1
0001656B <res 00000001>      <1> createfile_UpdatePDir:     resb 1 ; 31/03/2016
                             <1>
                             <1> ;alignb 4
                             <1>
                             <1> ; 11/04/2016
0001656C <res 00000002>      <1> env_var_length:       resw 1
                             <1>
0001656E <res 00000002>      <1> alignb 4
```

```
                             <1>
                             <1> ; 25/04/2016
00016570 <res 00000001>      <1> readi.valid: resb 1 ; valid data (>0 = valid for readi)
00016571 <res 00000001>      <1> readi.drv:   resb 1 ; drive number (0, 1,2,3,4..)
00016572 <res 00000001>      <1> readi.spc:   resb 1 ; sectors per cluster for 'readi' drive
00016573 <res 00000001>      <1> readi.s_index:  resb 1 ; sector index in current cluster (buffer)
00016574 <res 00000004>      <1> readi.sector:      resd 1 ; current disk sector
00016578 <res 00000002>      <1> readi.bpc:   resw 1 ; bytes per cluster - 1
0001657A <res 00000002>      <1> readi.offset:      resw 1 ; byte offset in cluster buffer
0001657C <res 00000004>      <1> readi.cluster:  resd 1 ; current cluster number
00016580 <res 00000004>      <1> readi.c_index:     resd 1 ; cluster index of the current cluster (0,1,2,3..)
00016584 <res 00000004>      <1> readi.fclust:      resd 1 ; first cluster of the current cluster
00016588 <res 00000004>      <1> readi.fs_index: resd 1 ; sector index in disk/file section (for Singlix FS)
                             <1> ;readi.buffer:     resd 1 ; readi sector buffer address
                             <1>
                             <1> ;alignb 4
                             <1>
0001658C <res 00000001>      <1> writei.valid:      resb 1 ; valid data (>0 = valid for writei)
0001658D <res 00000001>      <1> writei.drv:  resb 1 ; drive number (0, 1,2,3,4..)
0001658E <res 00000001>      <1> writei.spc:  resb 1 ; sectors per cluster for 'writei' drive
0001658F <res 00000001>      <1> writei.s_index: resb 1 ; sector index in current cluster (buffer)
00016590 <res 00000004>      <1> writei.sector:     resd 1 ; current disk sector
00016594 <res 00000002>      <1> writei.bpc:  resw 1 ; bytes per cluster - 1
00016596 <res 00000002>      <1> writei.offset:     resw 1 ; byte offset in cluster buffer
00016598 <res 00000004>      <1> writei.cluster: resd 1 ; current cluster number
0001659C <res 00000004>      <1> writei.c_index:    resd 1 ; cluster index of the current cluster (0,1,2,3..)
000165A0 <res 00000004>      <1> writei.fclust:  resd 1 ; first cluster of the current cluster
000165A4 <res 00000004>      <1> writei.fs_index: resd 1 ; sector index in disk/file section (for Singlix FS)
                             <1> ;writei.buffer:    resd 1 ; writei sector buffer address
000165A8 <res 00000004>      <1> writei.lclust:     resd 1 ; writei last cluster (mget_w) ; 23/10/2016
000165AC <res 00000004>      <1> writei.l_index:    resd 1 ; writei last cluster index (mget_w) ; 23/10/2016
000165B0 <res 00000001>      <1> writei.ofn:  resb 1 ; open file number (to be written) ; 23/10/2016
                             <1>
000165B1 <res 00000003>      <1> alignb 4
                             <1>
                             <1> ; 29/04/2016
000165B4 <res 00000004>      <1> Run_CDirFC:  resd 1
000165B8 <res 00000001>      <1> Run_Auto_Path:     resb 1
000165B9 <res 00000001>      <1> Run_Manual_Path: resb 1 ; 0 -> auto path sequence needed
000165BA <res 00000001>      <1> EXE_ID:            resb 1
000165BB <res 00000001>      <1> EXE_dot:     resb 1
                             <1>
                             <1> ; 06/05/2016
000165BC <res 00000004>      <1> mainprog_return_addr: resd 1
000165C0 <res 00000004>      <1> last_error:  resd 1  ; this will be used to return error code to MainProg
                             <1>                    ; 'lasterror' keyword will be used later to get the
                             <1>                    ; last error code/number/status.
                             <1> ; 12/05/2016
000165C4 <res 00000004>      <1> video_eax:   resd 1  ; eax return value of video function
                             <1>
                             <1> ; 01/06/2016
000165C8 <res 00000004>      <1> user_buffer: resd 1  ; 'diskio.s' (INT 33h, Function 08h, floppy disk type)
                             <1>
                             <1> ; 21/05/2016 - TRDOS 386 ('swap/switch', 'rswap', [u.pri])
000165CC <res 00000001>      <1> priority:    resb 1  ; running priority level of process (0,1,2)
                             <1>                    ; (run queue which is process comes from)
                             <1> ; 22/05/2016 - TRDOS 386 ('set_run_sequence', 'rtc_int', 'u_timer')
000165CD <res 00000001>      <1> p_change:    resb 1  ; process change status (for timer events)
                             <1> ; 23/05/2016 - TRDOS 386 ('clock')
000165CE <res 00000001>      <1> multi_tasking:     resb 1   ; Multi Tasking status (0 = disabled, >0 = enabled)
                             <1>                    ; (EBX will return with user buffer addr or disk type)
                             <1> ; 07/06/2016
000165CF <res 00000001>      <1> timer_events:      resb 1  ; number of (active) timer events, <= 16
                             <1>
                             <1> ; 24/06/2016
000165D0 <res 00000001>      <1> w_str_cmd:   resb 1 ; WRITE_STRING command (0,1,2,3) ; video.s
000165D1 <res 00000001>      <1> p_crt_mode:  resb 1  ; previous video mode (=3 or 0), backup mark/sign
                             <1> ; 26/06/2016
000165D2 <res 00000001>      <1> p_crt_page:  resb 1  ; previous active page (for 'set_mode')
                             <1> ; 04/07/2016
000165D3 <res 00000001>      <1> noclearmem:  resb 1  ; if set, 'SET MODE' (INT 31h) function (AH = 4)
                             <1>                    ; will not clear the video memory
                             <1>                    ; (usable for graphics modes only)
                             <1> alignb 2
000165D4 <res 00000002>      <1> CRT_LEN:     resw 1  ; length of regen buffer in bytes
000165D6 <res 00000010>      <1> cursor_pposn:      resw 8  ; cursor positions backup
                             <1>
                             <1> ; 10/07/2016 ('VGA_FONT_SETUP', INT 43H address for x86 real mode bios)
000165E6 <res 00000004>      <1> VGA_INT43H:  resd 1 ; 0 = default (not configured by user)
                             <1>                    ; 0FFFFFFFFh = user defined fonts
                             <1>                    ; address:
                             <1>                    ;      vgafont8
                             <1>                    ;      vgafont16
                             <1>                    ;      vgafont14
                             <1>
                             <1> ; 25/07/2016
000165EA <res 00000001>      <1> VGA_MTYPE:   resb 1  ; 0=CTEXT,1=MTEXT,2=CGA,3=PLANAR1,4=PLANAR4,5=LINEAR
                             <1>
                             <1> ; 23/10/2016
000165EB <res 00000001>      <1> setfmod            resb 1 ; update last modification date&time sign (if >0)
                             <1>                    ; (it is Open File Number + 1, if > 0)
                             <1> alignb 4
                             <1>
                             <1> ; 16/10/2016
000165EC <res 00000004>      <1> FFF_UBuffer: resd 1  ; User's buffer address for FFF & FNF system calls
                             <1> ; 15/10/2016
000165F0 <res 00000001>      <1> FFF_Valid:   resb 1  ; Find First File Structure validation byte
                             <1>                    ; 0  = invalid (Find Next File can't use FFF struct)
                             <1>                    ; >0 = valid, return type for FFF and Find Next File
                             <1>                    ; 24 = basic parameters, 24 bytes
                             <1>                    ; 128 = entire FFF structure/table, 128 bytes
                             <1> ; 16/10/2016 (FFF_Attrib: resw 1)
000165F1 <res 00000001>      <1> FFF_Attrib:  resb 1 ; Find First File attributes for Find Next File (LB)
```

```
000165F2 <res 00000001>      <1> FFF_RType:   resb 1  ; FFF return type (0 = Basic, >0 = complete) (HB)
                             <1> ; 16/10/2016 - 05/10/2016 (Set Working Path)
000165F3 <res 00000001>      <1> SWP_inv_fname:     resb 1 ; Set Working Path - Invalid File Name
000165F4 <res 00000002>      <1> SWP_Mode:    resw 1 ; Set Working Path - Mode
000165F6 <res 00000001>      <1> SWP_DRV:     resb 1 ; Set Working Path - Drive
000165F7 <res 00000001>      <1> SWP_DRV_chg: resb 1 ; Set Working Path - Drive Change
                             <1>
                             <1> ; 27/02/2017
000165F8 <res 00000001>      <1> fpready:     resb 1 ; '80387 fpu is ready' flag
                             <1>
                             <1> ; 08/10/2016
000165F9 <res 00000009>      <1> device_name:    resb 9  ; capitalized (and zero padded) device canem
                             <1>                    ; (example: "TTY0",0,0,0,0,0")
                             <1>
00016602 <res 00000002>      <1> alignb 4
                             <1>
                             <1> ; 08/10/2016
                             <1> ; 07/10/2016
                             <1> ; Table of kernel devices (which do not use installable device drivers)
                             <1> ; has been coded into KERNEL (trdosk9.s)
                             <1> ; 07/10/2016
                             <1> ; 8 installable device drivers available to install (NUMIDEV)
00016604 <res 00000020>      <1> IDEV_PGDIR: resd NUMIDEV
                             <1>                    ; Page directories of installable device drivers
                             <1>                    ;
                             <1>                    ; Note: Virtual start address is always 400000h
                             <1>                    ; (end of the 1st 4MB). [org 400000h]
                             <1>                    ; Segments: KCODE, KDATA
                             <1>                    ; Method: call 400000h (after changing page dir)
                             <1>                    ; Query code located at the start (400000h).
                             <1>                    ; Query code returns with
                             <1>                    ;   eax = device type and driver version
                             <1>                    ;         AL = Device Type minor
                             <1>                    ;         AH = Device Type major
                             <1>                    ;         Byte 16-23 : Version minor
                             <1>                    ;        Byte 24-31 : Version major - 1
                             <1>                    ;                   (0:0 -> 1.0)
                             <1>                    ;   ebx = initialization code address
                             <1>                    ;   ecx = configuration table address
                             <1>                    ;   edx = description table address
                             <1>                    ;   esi = device (default) name address (ASCIIZ)
                             <1>                    ;       (name has "/DEV/" prefix)
                             <1>                    ;   edi = dispatch table address
                             <1>                    ;        (for calling kernel-device functions)
                             <1>                    ;   ebp = address table address
                             <1>                    ; Initialization code returns with
                             <1>                    ;   eax = open code address
                             <1>                    ;   ecx = close code address
                             <1>                    ;   ebx = read code address
                             <1>                    ;   edx = write code address
                             <1>                    ;   esi = IOCTL code address
                             <1>                    ;   edi = dispatch table address
                             <1>                    ;   ebp = address table address
                             <1>                    ; Address Table:
                             <1>                    ;    Offset 0  : open code address
                             <1>                    ;    Offset 4  : read code address
                             <1>                    ;    Offset 8  : write code address
                             <1>                    ;    Offset 12 : close code address
                             <1>                    ;    Offset 16 : IOCTL code address
                             <1>                    ;    Offset 20 : initialization code address
                             <1>                    ;    Offset 24 : description table address
                             <1>                    ;    Offset 28 : configuration table address
                             <1>                    ;    Offset 32 : device name address
                             <1>                    ;    Offset 36 : dispatch table address
                             <1>                    ;          (for calling kernel-device functions)
                             <1>
00016624 <res 00000040>      <1> IDEV_NAME:  resb 8*NUMIDEV
                             <1>                    ; 8 byte names of installable device drivers
                             <1>
00016664 <res 00000008>      <1> IDEV_TYPE:  resb NUMIDEV ; Driver type of installable device drivers
0001666C <res 00000008>      <1> IDEV_FLAGS: resb NUMIDEV ; Device access parameters for installable
                             <1>                          ; device drivers (These values are set while
                             <1>                     ; the device driver is being loaded.)
00016674 <res 00000020>      <1> IDEV_OADDR: resd NUMIDEV ; open function addr for installable dev driver
00016694 <res 00000020>      <1> IDEV_CADDR: resd NUMIDEV ; close function addr for installable dev driver
000166B4 <res 00000020>      <1> IDEV_RADDR: resd NUMIDEV ; read function addr for installable dev driver
000166D4 <res 00000020>      <1> IDEV_WADDR: resd NUMIDEV ; write function addr for installable dev driver
                             <1>
                             <1> ; 08/10/2016
                             <1> ; 07/10/2016
                             <1> ; Device Open and Access parameters
000166F4 <res 0000001E>      <1> DEV_ACCESS:   resb NUMOFDEVICES   ; bit 0 = accessable by normal users
                             <1>                                ; bit 1 = read access permission
                             <1>                                ; bit 2 = write access permission
                             <1>                                ; bit 3 = IOCTL permission to users
                             <1>                                ; bit 4 = block device if it is set
                             <1>                                ; bit 5 = 16 bit or 1024 byte data
                             <1>                                ; bit 6 = 32 bit or 2048 byte data
                             <1>                                ; bit 7 = installable device driver
00016712 <res 0000001E>      <1> DEV_R_OWNER:  resb NUMOFDEVICES   ; Reading owner no (u.uid) of devices
00016730 <res 0000001E>      <1> DEV_R_OPENCOUNT: resb NUMOFDEVICES   ; Reading open count
0001674E <res 0000001E>      <1> DEV_W_OWNER:  resb NUMOFDEVICES   ; Writing owner no (u.uid) of devices
0001676C <res 0000001E>      <1> DEV_W_OPENCOUNT: resb NUMOFDEVICES   ; Writing open count
0001678A <res 0000001E>      <1> DEV_DRIVER:   resb NUMOFDEVICES   ; device driver number (1 to 7Fh)
                             <1>                                ; *if bit 7 is set (80 to FFh)
                             <1>                                ; *if it is installable device driver
                             <1>                                ; *index (0 to 7Fh)
                             <1>                                ; otherwise it is kernel device index
000167A8 <res 0000001E>      <1> DEV_OPENMODE:       resb NUMOFDEVICES   ; 1 = read mode
                             <1>                                ; 2 = write mode
                             <1>                                ; 3 = read & write
                             <1>                                ; 0 = not open (free)
000167C6 <res 00000078>      <1> DEV_NAME_PTR:    resd NUMOFDEVICES   ; pointers to name addresses of drivers
```

```
                                 <1>                                ; Address base: KDEV_NAME+
                                 <1>                                ; or IDEV_NAME+
0001683E <res 00000078>          <1> DEV_R_POINTER:      resd NUMOFDEVICES   ; reading pointer, writing pointer
000168B6 <res 00000078>          <1> DEV_W_POINTER:      resd NUMOFDEVICES   ; sector number if block device
                                 <1>                                ; character offset if char device
0001692E <res 00000002>          <1> alignb 4
                                 <1>
                                 <1> ; 06/10/2016
                                 <1> ; Open File Parameters
00016930 <res 00000028>          <1> OF_FCLUSTER: resd OPENFILES  ; First clusters of open files
00016958 <res 0000000A>          <1> OF_DRIVE:    resb OPENFILES  ; Logical DOS drive numbers of open files
00016962 <res 0000000A>          <1> OF_MODE:     resb OPENFILES  ; Open mode (1 = read, 2 = write, 3 = r&w)
0001696C <res 0000000A>          <1> OF_STATUS:   resb OPENFILES  ; (bit 0 = read, bit 1 = write)
00016976 <res 0000000A>          <1> OF_OPENCOUNT:      resb OPENFILES  ; Open counts of open files
00016980 <res 00000028>          <1> OF_POINTER:  resd OPENFILES      ; File seek/read/write pointer
000169A8 <res 00000028>          <1> OF_SIZE:     resd OPENFILES      ; File sizes of open files (in bytes)
000169D0 <res 00000028>          <1> OF_DIRFCLUSTER:    resd OPENFILES  ; Directory First Clusters of open files
000169F8 <res 00000028>          <1> OF_DIRCLUSTER:     resd OPENFILES  ; Directory (Entry) Clusters of open files
00016A20 <res 00000028>          <1> OF_VOLUMEID: resd OPENFILES  ; Vol ID for removable drives of open files
00016A48 <res 00000028>          <1> OF_CCLUSTER: resd OPENFILES  ; Current clusters of open files
00016A70 <res 00000028>          <1> OF_CCINDEX:  resd OPENFILES  ; Cluster index numbers of current clusters
                                 <1> ; 24/10/2016
00016A98 <res 00000014>          <1> OF_DIRENTRY: resw OPENFILES  ; Directory entry index no. in dir cluster
                                 <1>                           ; Sector index = entry index / 16
                                 <1> ;alignb 2
                                 <1>
00016AAC <res 00000060>          <1> DTA:         resd 24             ; Find First File data transfer area
                                 <1>
                                 <1> ; 19/12/2016
00016B0C <res 00000001>          <1> tcallback:   resb 1       ; Timer callback method flag for 'systimer'
00016B0D <res 00000001>          <1> trtc:        resb 1       ; Timer interrupt type flag for 'systimer'
                                 <1> ; 20/02/2017
00016B0E <res 00000001>          <1> no_page_swap:      resb 1       ; Swap lock for Signal Response Byte pages
                                 <1> ;;15/01/2017
                                 <1> ; 02/01/2017
                                 <1> ;;intflg:    resb 1       ; software interrupt in progress signal
                                 <1>                           ; (for timer interrupt)
                                 <1>
00016B0F <res 00000001>          <1> alignb 4
                                 <1> ; 13/04/2017
00016B10 <res 0000001E>          <1> DEV_INTR:    resb NUMOFDEVICES ; Device Interrupt (IRQ) number + 1
                                 <1>                           ; (0= not available, 1= IRQ 0, 16= IRQ 15)
00016B2E <res 00000040>          <1> DEV_INT_HNDLR:     resd 16             ; Device Interrupt Handler addr, if > 0
                                 <1>
                                 <1>
                                 <1> ;alignb 4
                                 <1>
                                 <1> ; 26/02/2017 ; IRQ Callback parameters ('syscalbac')
                                 <1> ;Index: ; 0 to 8
                                 <1> ;     0 = IRQ3, 1 = IRQ4, 2 = IRQ5, 3 = IRQ7
                                 <1> ;     4 = IRQ9, 5 = IRQ10, 6 = IRQ11, 7 = IRQ12, 8 = IRQ13
00016B6E <res 00000009>          <1> IRQ.owner:   resb 9       ; owner, 0 = free, >0 = [u.uno]
00016B77 <res 00000009>          <1> IRQ.dev:     resb 9       ; 0 = default/kernel, >0 = device number
00016B80 <res 00000009>          <1> IRQ.method:  resb 9              ; 0 = Signal Response Byte, 1 = Callback
00016B89 <res 00000009>          <1> IRQ.srb:     resb 9              ; Signal Response/Return Byte value
00016B92 <res 00000024>          <1> IRQ.addr:    resd 9       ; Rignal Response Byte address (physical)
                                 <1>                           ; or Callback service address (virtual)
                                 <1> ; 28/02/2017
00016BB6 <res 00000004>          <1> IRQ_cr3:     resd 1       ; for saving cr3 register in IRQ handler
00016BBA <res 00000001>          <1> IRQnum:            resb 1       ; IRQ number for IRQ handler (trdosk8.s)
                                 <1>
                                 <1> ; 10/04/2017
                                 <1> ; 03/04/2017
                                 <1> ; UNINITIALIZED AUDIO DATA
00016BBB <res 00000001>          <1> alignb 4
00016BBC <res 00000001>          <1> audio_pci:   resb 1
00016BBD <res 00000001>          <1> audio_device:      resb 1
00016BBE <res 00000001>          <1> audio_mode:  resb 1
00016BBF <res 00000001>          <1> audio_intr:  resb 1
00016BC0 <res 00000001>          <1> audio_busy:  resb 1  ; Busy flag for audio irq ; 21/04/2017
00016BC1 <res 00000001>          <1> audio_reserved: resb 1
00016BC2 <res 00000002>          <1> audio_io_base:     resw 1       ; Base I/O address of audio device
00016BC4 <res 00000004>          <1> audio_dev_id:      resd 1 ; BUS/DEV/FN ; 00000000BBBBBBBBDDDDDFFF00000000
00016BC8 <res 00000004>          <1> audio_vendor:      resd 1
00016BCC <res 00000004>          <1> audio_stats_cmd: resd 1
                                 <1> ;
00016BD0 <res 00000004>          <1> audio_buffer:      resd 1 ; virtual address of user's audio buffer
00016BD4 <res 00000004>          <1> audio_p_buffer:    resd 1 ; Physical address of user's audio buffer
00016BD8 <res 00000004>          <1> audio_buff_size: resd 1 ; user's audio buffer size (half buffer size)
00016BDC <res 00000004>          <1> audio_dma_buff: resd 1  ; dma buffer address
00016BE0 <res 00000004>          <1> audio_dmabuff_size: resd 1 ; dma buffer size (2 * half buffer size)
00016BE4 <res 00000001>          <1> audio_flag:  resb 1  ; dma buffer flag (1st half = 0, 2nd half = 1)
00016BE5 <res 00000001>          <1> audio_user:  resb 1 ; user number of the owner
00016BE6 <res 00000001>          <1> audio_cb_mode:     resb 1 ; 0 = signal response byte method
                                 <1>                    ; 1 = callback method
                                 <1>                    ; 2 = s.r.b. method with auto increment
00016BE7 <res 00000001>          <1> audio_srb:   resb 1 ; signal response byte value
00016BE8 <res 00000004>          <1> audio_cb_addr:     resd 1  ; callback service address or s.r.b. address
                                 <1>                    ; (s.r.b. addr is physical, cbs addr is virtual)
                                 <1>
00016BEC <res 00000001>          <1> audio_bps:   resb 1  ; selected mode: 8 bit, 16 bit
00016BED <res 00000001>          <1> audio_stmo:  resb 1 ; selected mode: mono /stereo
00016BEE <res 00000002>          <1> audio_freq:  resw 1 ; sampling rate
                                 <1>
                                 <1> ; 21/04/2017
00016BF0 <res 00000001>          <1> audio_play_cmd: resb 1  ; Play/Stop command (1 = play, 0 = stop)
                                 <1> audio_civ: ; 28/05/2017 ; Current Buffer Index (AC'97)
00016BF1 <res 00000001>          <1> audio_flag_eol:    resb 1  ; End of Link status (vt8233, EOL/FLAG)
                                 <1>
                                 <1> audio_master_volume:
00016BF2 <res 00000001>          <1> audio_master_volume_l: resb 1 ; sound volume (lineout) left channel
00016BF3 <res 00000001>          <1> audio_master_volume_r: resb 1 ; sound volume (lineout) right channel
                                 <1>
```

```
                               <1> alignb 4
                               <1> ; 28/05/2017
                               <1> ; AC'97 Audio Controller Base Adress Registers
00016BF4 <res 00000002>        <1> NAMBAR:          resw 1 ; Native Audio Mixer Base Address
00016BF6 <res 00000002>        <1> NABMBAR:    resw 1 ; Native Audio Bus Mastering Base Address
                               <1>
                               <1> ;alignb 4
                               <1> ; 21/04/2017
00016BF8 <res 00000400>        <1> audio_bdl_buff:    resd 32*8  ; VT8233 (AC97) BDL Buffer Size
                               <1> ; 12/05/2017
00016FF8 <res 00000004>        <1> base_addr:  resd 1 ; 'direct_memory_access' (memory.s)
                               <1>
                               <1> ; 28/08/2017
                               <1> ; 20/08/2017
00016FFC <res 00000001>        <1>             resb 1  ;
00016FFD <res 00000001>        <1> dma_user:   resb 1 ; user number for sysdma
00016FFE <res 00000001>        <1> dma_channel: resb 1 ; dma channel for sysdma
00016FFF <res 00000001>        <1> dma_mode:   resb 1  ; dma mode for sysdma
00017000 <res 00000004>        <1> dma_addr:   resd 1 ; dma buffer physical addr for sysdma
00017004 <res 00000004>        <1> dma_size:   resd 1  ; dma buffer size (in bytes) for sysdma
00017008 <res 00000004>        <1> dma_start:  resd 1  ; dma start address for sysdma
0001700C <res 00000004>        <1> dma_count:  resd 1  ; dma count (in bytes) for sysdma
                               <1>
00017010 <res 00008FF0>        <1> alignb 65536
                               <1> ; 09/08/2017
                               <1> ; 12/05/2017
00020000 <res 00010000>        <1> sb16_dma_buffer: resb 65536 ; DMA buffer for sb16 audio playing.
                                   ; 24/01/2016
                                   %include 'ubss.s'  ; UNINITIALIZED KERNEL (USER) DATA
                               <1> ; *****************************************************************************
                               <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - UNINITIALIZED USER DATA : ubss.s
                               <1> ; -----------------------------------------------------------------------------
                               <1> ; Last Update: 28/02/2017
                               <1> ; -----------------------------------------------------------------------------
                               <1> ; Beginning: 24/01/2016
                               <1> ; -----------------------------------------------------------------------------
                               <1> ; Assembler: NASM version 2.11 (trdos386.s)
                               <1> ; -----------------------------------------------------------------------------
                               <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
                               <1> ; ux.s (04/12/2015)
                               <1> ; *****************************************************************************
                               <1>
                               <1> ; Retro UNIX 386 v1 Kernel - ux.s
                               <1> ; Last Modification: 04/12/2015
                               <1> ;
                               <1> ; ///////// RETRO UNIX 386 V1 SYSTEM DEFINITIONS //////////////
                               <1> ; (Modified from
                               <1> ;      Retro UNIX 8086 v1 system definitions in 'UNIX.ASM', 01/09/2014)
                               <1> ; ((UNIX.ASM (RETRO UNIX 8086 V1 Kernel), 11/03/2013 - 01/09/2014))
                               <1> ; -----------------------------------------------------------------------------
                               <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
                               <1> ; (Original) Source Code by Ken Thompson (1971-1972)
                               <1> ; <Bell Laboratories (17/3/1972)>
                               <1> ; <Preliminary Release of UNIX Implementation Document>
                               <1> ; (Section E10 (17/3/1972) - ux.s)
                               <1> ; *****************************************************************************
                               <1>
                               <1> alignb 2
                               <1>
                               <1> inode:
                               <1>       ; 11/03/2013.
                               <1>       ;Derived from UNIX v1 source code 'inode' structure (ux).
                               <1>       ;i.
                               <1>
00030000 <res 00000002>        <1>       i.flgs:       resw 1
00030002 <res 00000001>        <1>       i.nlks:       resb 1
00030003 <res 00000001>        <1>       i.uid:  resb 1
                               <1>         ;i.size:  resw 1 ; size
00030004 <res 00000002>        <1>       resw 1 ; 29/04/2016
00030006 <res 00000010>        <1>       i.dskp:       resw 8 ; 16 bytes
00030016 <res 00000004>        <1>       i.ctim:       resd 1
0003001A <res 00000004>        <1>       i.mtim:       resd 1
0003001E <res 00000002>        <1>       i.rsvd:  resw 1 ; Reserved (ZERO/Undefined word for UNIX v1.)
                               <1>
                               <1> I_SIZE      equ $ - inode
                               <1>
                               <1> process:
                               <1>       ; 19/12/2016
                               <1>       ; 21/05/2016
                               <1>       ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
                               <1>       ; 06/05/2015 - Retro UNIX 386 v1
                               <1>       ; 11/03/2013 - 05/02/2014 (Retro UNIX 8086 v1)
                               <1>       ;Derived from UNIX v1 source code 'proc' structure (ux).
                               <1>       ;p.
                               <1>
00030020 <res 00000020>        <1>         p.pid:   resw nproc
00030040 <res 00000020>        <1>         p.ppid:  resw nproc
00030060 <res 00000020>        <1>         p.break: resw nproc
00030080 <res 00000010>        <1>         p.ttyc:  resb nproc ; console tty in Retro UNIX 8086 v1.
00030090 <res 00000010>        <1>       p.waitc: resb nproc ; waiting channel in Retro UNIX 8086 v1.
000300A0 <res 00000010>        <1>       p.link:       resb nproc
000300B0 <res 00000010>        <1>       p.stat:       resb nproc
                               <1>
                               <1>       ; 06/05/2015 (Retro UNIX 386 v1 feature only !)
000300C0 <res 00000040>        <1>       p.upage: resd nproc ; Physical address of the process's
                               <1>                          ; 'user' structure
                               <1>       ; 21/05/2016
                               <1>       ; 19/05/2016 (TRDOS 386 feature only!)
00030100 <res 00000010>        <1>       p.timer: resb nproc ; number of timer events of the processs
                               <1>
                               <1>       ; 19/12/2016
00030110 <res 00000040>        <1>       p.tcb: resd nproc ; timer callback service address (if > 0)
                               <1>
```

```
                             <1> P_SIZE      equ $ - process
                             <1> 
                             <1> ; fsp table (original UNIX v1)
                             <1> ;
                             <1> ;Entry
                             <1> ;          15                                      0
                             <1> ;  1     |---|---------------------------------------|
                             <1> ;        |r/w|       i-number of open file           |
                             <1> ;        |---|---------------------------------------|
                             <1> ;        |               device number               |
                             <1> ;        |-------------------------------------------|
                             <1> ;    (*) | offset pointer, i.e., r/w pointer to file |
                             <1> ;        |-------------------------------------------|
                             <1> ;        |  flag that says    | number of processes  |
                             <1> ;        |   file deleted     | that have file open  |
                             <1> ;        |-------------------------------------------|
                             <1> ;  2     |                                           |
                             <1> ;        |-------------------------------------------|
                             <1> ;        |                                           |
                             <1> ;        |-------------------------------------------|
                             <1> ;        |                                           |
                             <1> ;        |-------------------------------------------|
                             <1> ;        |                                           |
                             <1> ;        |-------------------------------------------|
                             <1> ;  3     |                                           |
                             <1> ;        |                                           |
                             <1> ;
                             <1> ; (*) Retro UNIX 386 v1 modification: 32 bit offset pointer
                             <1> 
                             <1> 
                             <1> ; 15/04/2015
00030150 <res 000001F4>      <1> fsp:   resb nfiles * 10 ; 11/05/2015 (8 -> 10)
00030344 <res 00000002>      <1> idev:  resw 1 ; device number is 1 byte in Retro UNIX 8086 v1 !
00030346 <res 00000002>      <1> cdev:    resw 1 ; device number is 1 byte in Retro UNIX 8086 v1 !
                             <1> ; 18/05/2015
                             <1> ; 26/04/2013 device/drive parameters (Retro UNIX 8086 v1 feature only!)
                             <1> ; 'UNIX' device numbers (as in 'cdev' and 'u.cdrv')
                             <1> ;     0 -> root device (which has Retro UNIX 8086 v1 file system)
                             <1> ;     1 -> mounted device (which has Retro UNIX 8086 v1 file system)
                             <1> ; 'Retro UNIX 8086 v1' device numbers: (for disk I/O procedures)
                             <1> ;     0 -> fd0 (physical drive, floppy disk 1), physical drive number = 0
                             <1> ;     1 -> fd1 (physical drive, floppy disk 2), physical drive number = 1
                             <1> ;     2 -> hd0 (physical drive, hard disk 1), physical drive number = 80h
                             <1> ;     3 -> hd1 (physical drive, hard disk 2), physical drive number = 81h
                             <1> ;     4 -> hd2 (physical drive, hard disk 3), physical drive number = 82h
                             <1> ;     5 -> hd3 (physical drive, hard disk 4), physical drive number = 83h
00030348 <res 00000001>      <1> rdev:  resb 1 ; root device number ; Retro UNIX 8086 v1 feature only!
                             <1>               ; as above, for physical drives numbers in following table
00030349 <res 00000001>      <1> mdev:  resb 1 ; mounted device number ; Retro UNIX 8086 v1 feature only!
                             <1> ; 15/04/2015
0003034A <res 00000001>      <1> active:      resb 1
0003034B <res 00000001>      <1>        resb 1 ; 09/06/2015
0003034C <res 00000002>      <1> mnti:  resw 1
0003034E <res 00000002>      <1> mpid:  resw 1
00030350 <res 00000002>      <1> rootdir: resw 1
                             <1> 
                             <1> ; 21/05/2016 - TRDOS 386 (TRDOS v2.0) - priority levels, 3 run queues
                             <1> runq:
00030352 <res 00000002>      <1> runq_event:  resw 1 ; high priority, 'run for event'            ; 2
00030354 <res 00000002>      <1> runq_normal: resw 1 ; normal/regular priority, 'run as reqular' ; 1
00030356 <res 00000002>      <1> runq_background: resw 1 ; low priority, 'run on background'         ; 0
                             <1> ;
00030358 <res 00000001>      <1> imod:  resb 1
00030359 <res 00000001>      <1> smod:  resb 1
0003035A <res 00000001>      <1> mmod:  resb 1
0003035B <res 00000001>      <1> sysflg:      resb 1
                             <1> 
                             <1> alignb 4
                             <1> 
                             <1> user:
                             <1>       ; 13/01/2017
                             <1>       ; 19/12/2016
                             <1>       ; 21/05/2016 - TRDOS 386 (TRDOS v2.0)
                             <1>       ;             [u.pri] usage method modification
                             <1>       ; 04/12/2015
                             <1>       ; 18/10/2015
                             <1>       ; 12/10/2015
                             <1>       ; 21/09/2015
                             <1>       ; 24/07/2015
                             <1>       ; 16/06/2015
                             <1>       ; 09/06/2015
                             <1>       ; 11/05/2015
                             <1>       ; 16/04/2015 (Retro UNIX 386 v1 - 32 bit modifications)
                             <1>       ; 10/10/2013
                             <1>       ; 11/03/2013.
                             <1>       ;Derived from UNIX v1 source code 'user' structure (ux).
                             <1>       ;u.
                             <1> 
0003035C <res 00000004>      <1>       u.sp:   resd 1 ; esp (kernel stack at the beginning of 'sysent')
00030360 <res 00000004>      <1>       u.usp:  resd 1 ; esp (kernel stack points to user's registers)
00030364 <res 00000004>      <1>       u.r0:   resd 1 ; eax
00030368 <res 00000002>      <1>       u.cdir:        resw 1
0003036A <res 0000000A>      <1>       u.fp:   resb 10
00030374 <res 00000004>      <1>       u.fofp:        resd 1
00030378 <res 00000004>      <1>       u.dirp:        resd 1
0003037C <res 00000004>      <1>       u.namep:  resd 1
00030380 <res 00000004>      <1>       u.off:  resd 1
00030384 <res 00000004>      <1>       u.base:        resd 1
00030388 <res 00000004>      <1>       u.count:  resd 1
0003038C <res 00000004>      <1>       u.nread:  resd 1
00030390 <res 00000004>      <1>       u.break:  resd 1 ; break
00030394 <res 00000002>      <1>       u.ttyp:        resw 1
                             <1>       ; 10/01/2017 (TRDOS 386, relocation and dword alignment)
```

```
                                <1>      ; tty number (rtty, rcvt, wtty)
00030396 <res 00000001>         <1>      u.ttyn:       resb 1 ; 28/07/2013 - Retro Unix 8086 v1 feature only !
00030397 <res 00000001>         <1>      u.resb:   resb 1 ; 10/01/2017 (TRDOS 386, temporary)
00030398 <res 00000010>         <1>      u.dirbuf: resb 16 ; 04/12/2015 (10 -> 16)
                                <1>      ;u.pri:       resw 1 ; 14/02/2014
000303A8 <res 00000001>         <1>      u.quant:  resb 1 ; Retro UNIX 8086 v1 Feature only ! (uquant)
000303A9 <res 00000001>         <1>      u.pri:   resb 1 ; Modification: 21/05/2016 (priority levels: 0, 1, 2)
000303AA <res 00000002>         <1>      u.intr:       resw 1
000303AC <res 00000002>         <1>      u.quit:       resw 1
                                <1>      ;u.emt:       resw 1 ; 10/10/2013
                                <1>      ;u.ilgins: resw 1 ; 10/01/2017
000303AE <res 00000002>         <1>      u.cdrv:       resw 1 ; cdev
000303B0 <res 00000001>         <1>      u.uid:   resb 1 ; uid
000303B1 <res 00000001>         <1>      u.ruid:       resb 1
000303B2 <res 00000001>         <1>      u.bsys:       resb 1
000303B3 <res 00000001>         <1>      u.uno:   resb 1
000303B4 <res 00000004>         <1>        u.upage:  resd 1 ; 16/04/2015 - Retro Unix 386 v1 feature only !
000303B8 <res 00000004>         <1>      u.pgdir:  resd 1 ; 09/03/2015 (page dir addr of process)
000303BC <res 00000004>         <1>      u.ppgdir: resd 1 ; 06/05/2015 (page dir addr of the parent process)
000303C0 <res 00000004>         <1>      u.pbase:  resd 1 ; 20/05/2015 (physical base/transfer address)
000303C4 <res 00000002>         <1>      u.pcount: resw 1 ; 20/05/2015 (byte -transfer- count for page)
                                <1>      ;u.pncount: resw 1
                                <1>             ; 16/06/2015 (byte -transfer- count for page, 'namei', 'mkdir')
                                <1>      ;u.pnbase:  resd 1
                                <1>             ; 16/06/2015 (physical base/transfer address, 'namei', 'mkdir')
                                <1>                     ; 09/06/2015
000303C6 <res 00000001>         <1>      u.kcall:  resb 1 ; The caller is 'namei' (dskr) or 'mkdir' (dskw) sign
000303C7 <res 00000001>         <1>      u.brwdev: resb 1 ; Block device number for direct I/O (bread & bwrite)
                                <1>                     ; 24/07/2015 - 24/06/2015
                                <1>      ;u.args:  resd 1 ; arguments list (line) offset from start of [u.upage]
                                <1>                     ; (arg list/line is from offset [u.args] to 4096 in [u.upage])
                                <1>                     ; ([u.args] points to argument count -argc- address offset)
                                <1>                     ; 24/06/2015
                                <1>      ;u.core:  resd 1 ; physical start address of user's memory space (for sys exec)
                                <1>      ;u.ecore: resd 1 ; physical end address of user's memory space (for sys exec)
                                <1>      ; last error number
000303C8 <res 00000004>         <1>      u.error:  resd 1 ; 28/07/2013 - 09/03/2015
                                <1>                     ; Retro UNIX 8086/386 v1 feature only!
                                <1>                    ; 21/09/2015 (debugging - page fault analyze)
000303CC <res 00000004>         <1>      u.pfcount: resd 1 ; page fault count for (this) process (for sys geterr)
                                <1>             ; 19/12/2016 (TRDOS 386)
000303D0 <res 00000004>         <1>      u.tcb:   resd 1 ; Timer callback address/flag which will be used by timer int
                                <1>             ; 13/01/2017 (TRDOS 386)
000303D4 <res 00000001>         <1>      u.t_lock: resb 1 ; Timer interrupt (callback) lock (unlocked by 'sysrele')
000303D5 <res 00000001>         <1>      u.t_mode: resb 1 ; running mode during timer interrupt (0= system, 0FFh= user)
                                <1>             ; 26/02/2017 (TRDOS 386)
000303D6 <res 00000001>         <1>      u.irqc:       resb 1  ; Count of IRQ callback services (IRQs in use)
                                <1>             ; 28/02/2017 (TRDOS 386)
000303D7 <res 00000001>         <1>      u.irqwait: resb 1 ; IRQ waiting for callback service flag (IRQ number, If > 0)
000303D8 <res 00000001>         <1>      u.r_lock:  resb 1 ; 'IRQ callback service is in progress' flag (IRQ lock)
000303D9 <res 00000001>         <1>      u.r_mode:  resb 1 ; running mode during hadware interrupt
                                <1>             ; 27/02/2017 (TRDOS 386)
000303DA <res 00000001>         <1>      u.fpsave: resb 1  ; TRDOS 386, 'save/restore FPU registers' flag
000303DB <res 00000001>         <1> alignb 4
000303DC <res 0000005E>         <1>      u.fpregs: resb 94 ; 94 byte area for saving and restoring FPU registers
                                <1>
0003043A <res 00000002>         <1> alignb 4
                                <1>
                                <1> U_SIZE      equ $ - user
                                <1>
                                <1> ; 18/10/2015 - Retro UNIX 386 v1 (local variables for 'namei' and 'sysexec')
0003043C <res 00000004>         <1> pcore:  resd 1  ; physical start address of user's memory space (for sys exec)
00030440 <res 00000004>         <1> ecore:  resd 1  ; physical address of user's stack/last page (for sys exec)
00030444 <res 00000004>         <1> nbase:       resd 1 ; physical base address for 'namei' & 'sysexec'
00030448 <res 00000002>         <1> ncount: resw 1     ; remain byte count in page for 'namei' & 'sysexec'
0003044A <res 00000002>         <1> argc: resw 1 ; argument count for 'sysexec'
0003044C <res 00000004>         <1> argv: resd 1 ; argument list (recent) address for 'sysexec'
                                <1>
                                <1> ; 03/06/2015 - Retro UNIX 386 v1 Beginning
                                <1> ; 07/04/2013 - 31/07/2013 - Retro UNIX 8086 v1
00030450 <res 00000001>         <1> rw:    resb 1 ;; Read/Write sign (iget)
                                <1>
                                <1> ;alignb 4
                                <1>
                                <1> ; 24/04/2016
00030451 <res 00000004>         <1> ii:         resd 1 ; first cluster of the program file
00030455 <res 00000004>         <1> i.size:           resd 1 ; size of the program file

00030459 <res 00000003>             alignb 4

                                    ; 23/05/2016 (TRDOS 386)
                                    ; 14/10/2015 (Retro UNIX 386 v1, 'unix386.s')
0003045C <res 00000004>             cr3reg:      resd 1  ; cr3 register content at the beginning of the timer
                                                 ; (or RTC) interrupt handler.

                                    ; 10/12/2016 (callback)
                                    ; 10/06/2016
                                    ; 19/05/2016
                                    ; 18/05/2016 - TRDOS 386 feature only !
00030460 <res 00000100>             timer_set: resd 16*4   ; 256 bytes memory space for 16 timer events
                                          ; Timer Event Structure: (max. 16 timer events, 16*16 bytes)
                                          ;       Owner:              resb 1 ; 0 = free
                                          ;                          ;>0 = process number (u.uno)
                                          ;      Callback:    resb 1 ; 0 = response byte address (phy)
                                          ;                           1 = callback address (virtual)
                                          ;      Interrupt:      resb 1 ; 0 = Timer interrupt (or none)
                                          ;                          ; 1 = Real Time Clock interrupt
                                          ;      Response:       resb 1 ; 0 to 255, signal return value
                                          ;      Count Limit: resd 1 ; count of ticks (total/set)
                                          ;      Current Count:      resd 1 ; count of ticks (current)
                                          ;      Response Addr:  resd 1 ; response byte (pointer) address
                                          ;                          ; (or callback -user service- address)
```

```
                                     ;; Memory (swap) Data (11/03/2015)
                                     ; 09/03/2015
00030560 <res 00000002>              swpq_count: resw 1 ; count of pages on the swap queue
00030562 <res 00000004>              swp_drv:    resd 1 ; logical drive description table address of the swap drive/disk
00030566 <res 00000004>              swpd_size:  resd 1 ; size of swap drive/disk (volume) in sectors (512 bytes).

0003056A <res 00000004>              swpd_free:  resd 1 ; free page blocks (4096 bytes) on swap disk/drive (logical)
0003056E <res 00000004>              swpd_next:  resd 1 ; next free page block
00030572 <res 00000004>              swpd_last:  resd 1 ; last swap page block

00030576 <res 00000002>              alignb 4

                                     ; 10/07/2015
                                     ; 28/08/2014
00030578 <res 00000004>              error_code: resd 1
                                     ; 29/08/2014
0003057C <res 00000004>              FaultOffset:       resd 1
                                     ; 21/09/2015
00030580 <res 00000004>              PF_Count:   resd 1 ; total page fault count
                                                              ; (for debugging - page fault analyze)
                                                        ; 'page_fault_handler' (memory.inc)
                                                        ; 'sysgeterr' (u9.s)

                                     ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)
                                     ; 22/08/2015 (Retro UNIX 386 v1)
                                     buffer:
00030584 <res 00000008>                    resb   8
                                     readi_buffer:
0003058C <res 00000200>                    resb   512
0003078C <res 00000008>                    resb   8
                                     writei_buffer:
00030794 <res 00000200>                    resb   512
                                     ; 24/10/2016
00030994 <res 00000008>                    resb   8
                                     rw_buffer:
0003099C <res 00000800>                    resb   2048  ; general purposed, r/w sector buffer

                                     bss_end:

                                     ; 27/12/2013
                                     _end:  ; end of kernel code
```